@rmoff #kafkasummit

From Zero to Hero with Kafka Connect

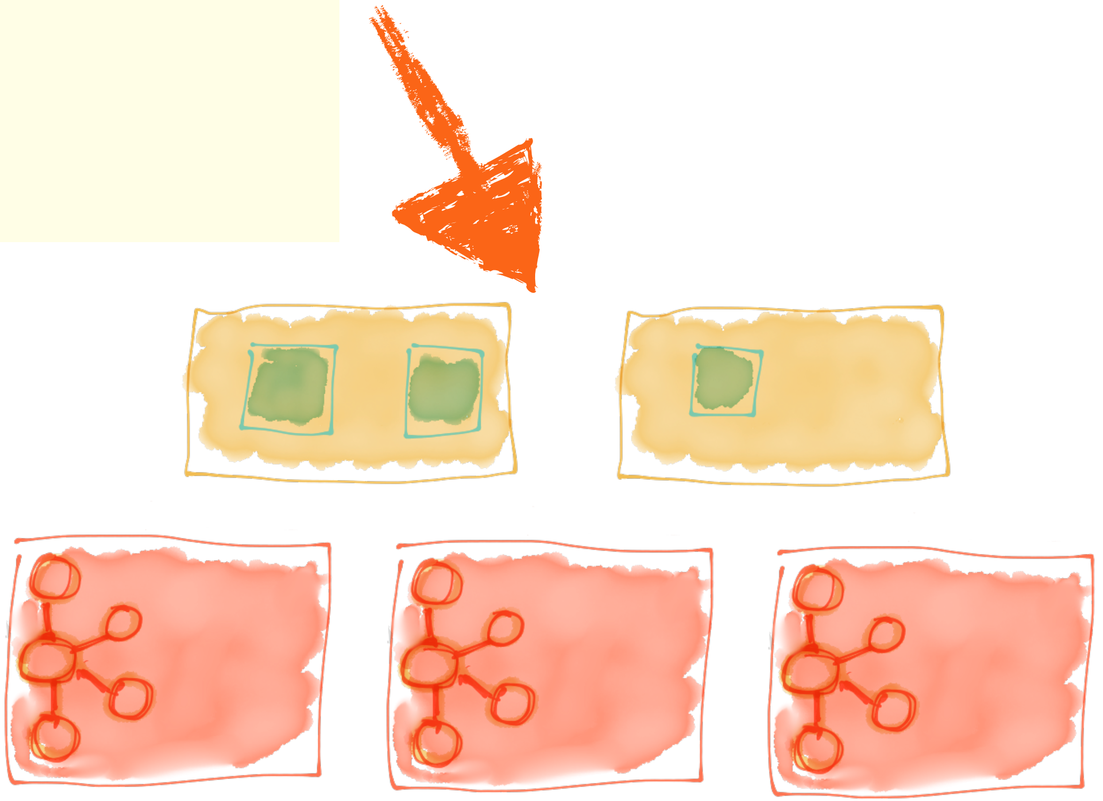
a.k.a.

A practical guide to becoming l33t with Kafka Connect

What is Kafka Connect?

Streaming Integration with Kafka Connect

Sources

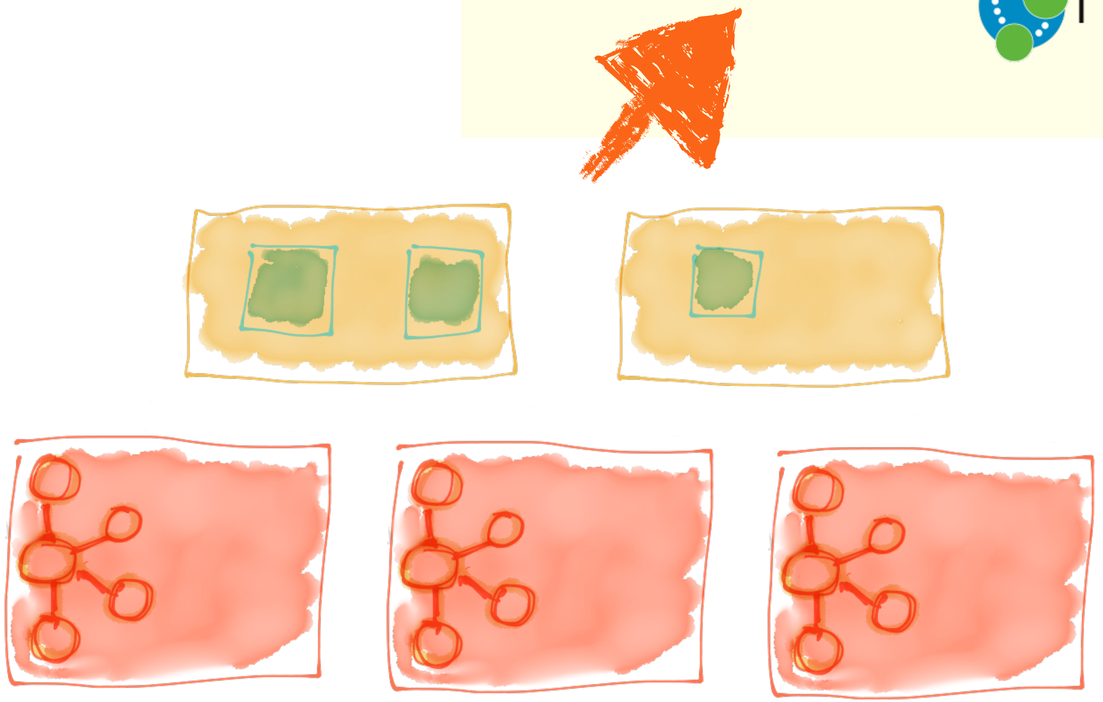


Kafka Connect

Kafka Brokers

Streaming Integration with Kafka Connect

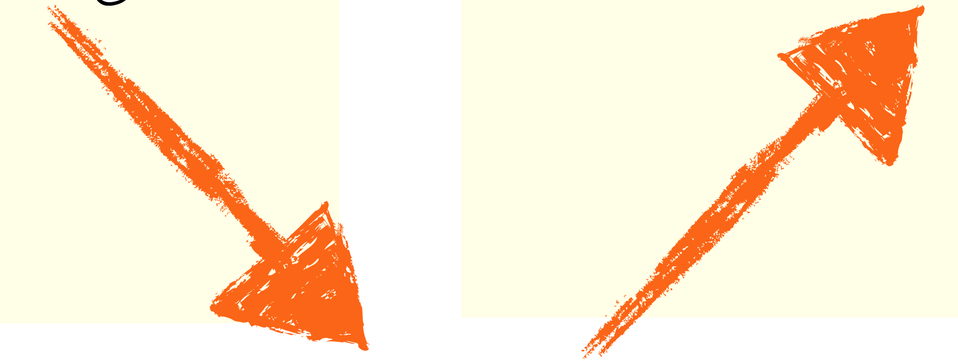
Sinks

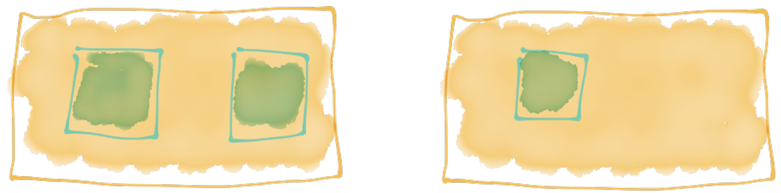


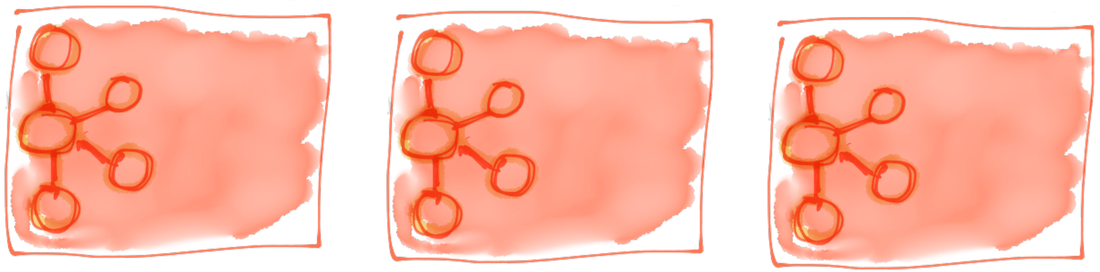
Kafka Connect

Kafka Brokers

Streaming Integration with Kafka Connect







Kafka Connect

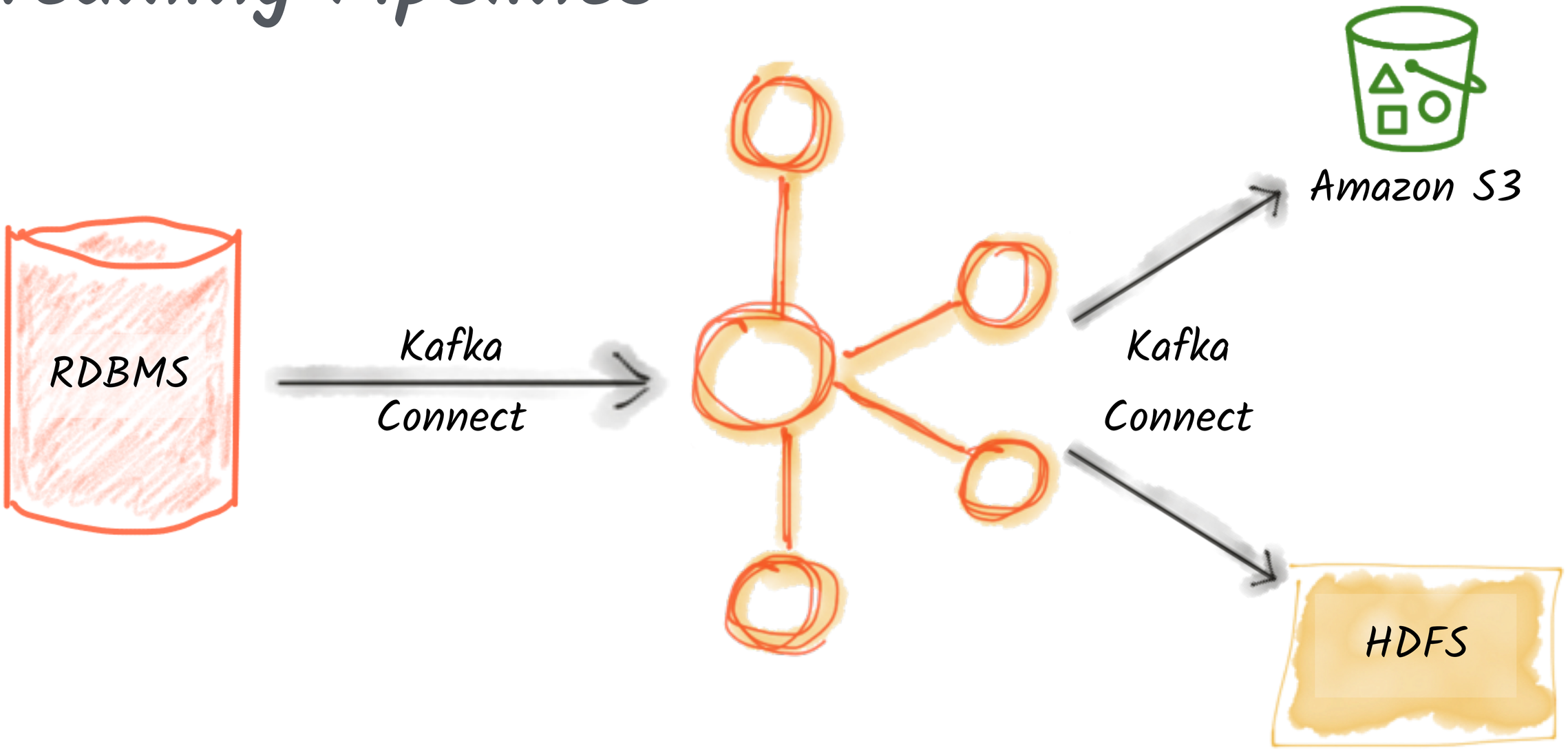
Kafka Brokers

Look Ma, No Code!

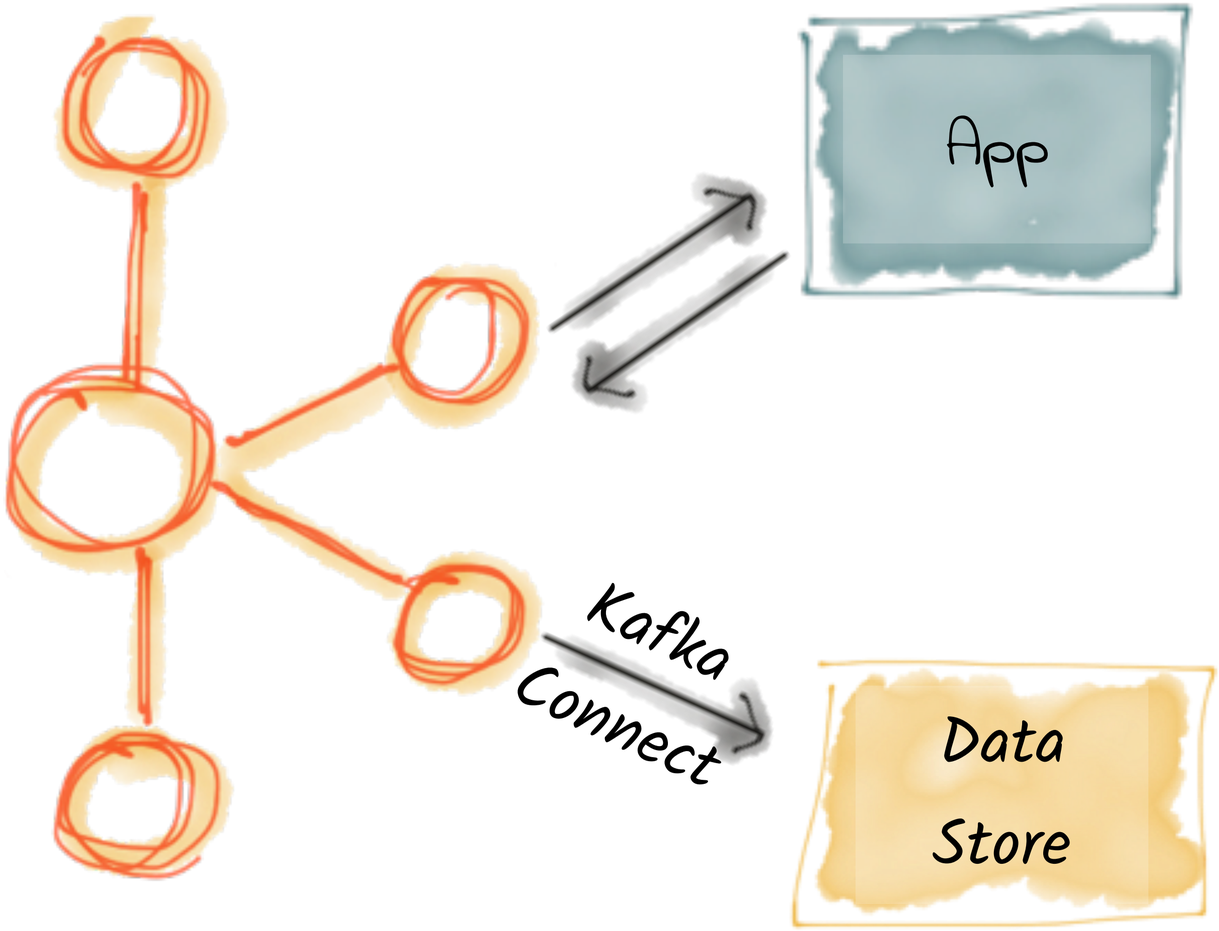
```
{
  "connector.class":
    "io.confluent.connect.jdbc.JdbcSourceConnector",
  "connection.url":
    "jdbc:mysql://asgard:3306/demo",
  "table.whitelist":
    "sales,orders,customers"
}
```

https://docs.confluent.io/current/connect/

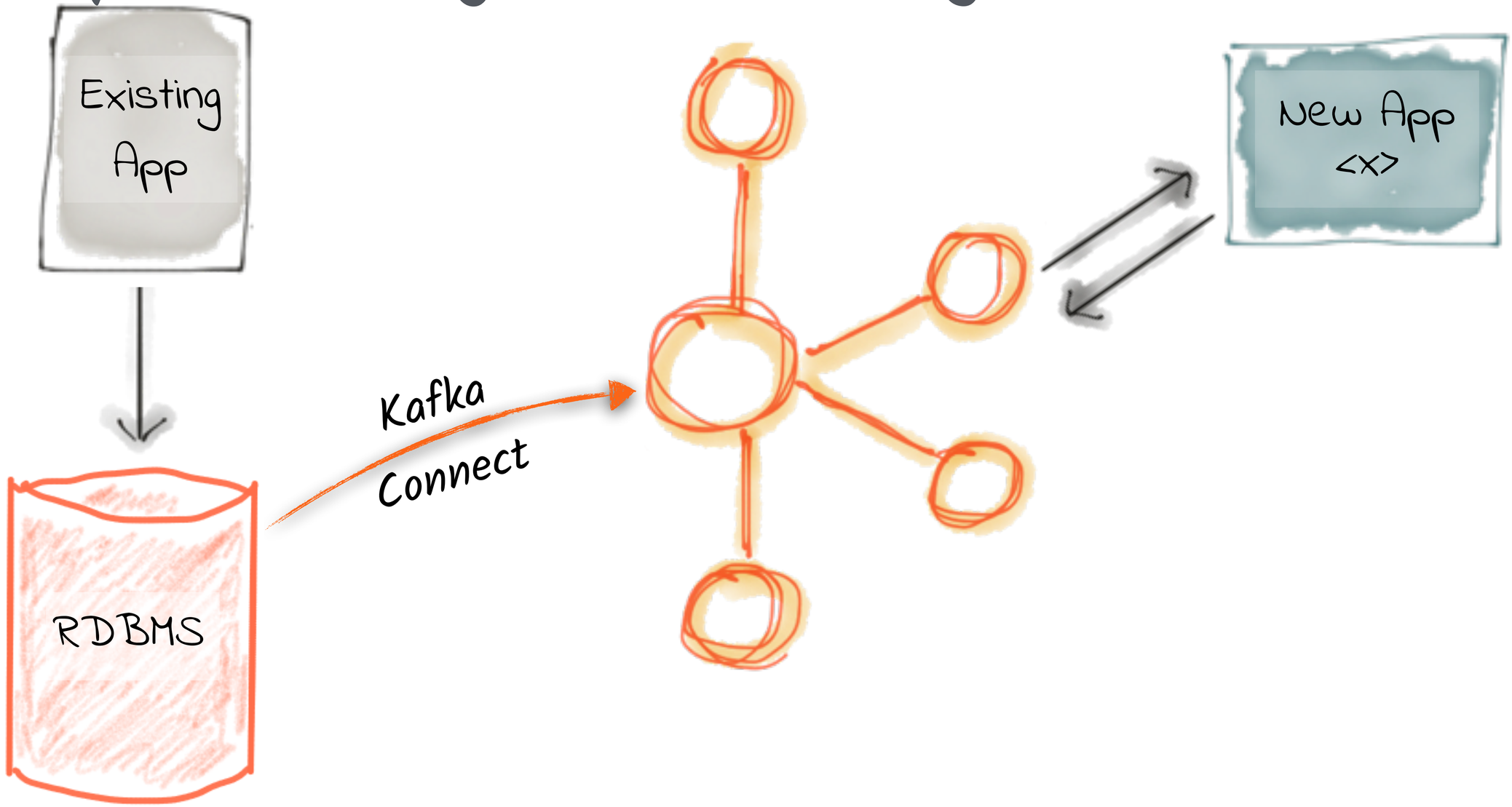
Streaming Pipelines



Writing to data stores from Kafka



Evolve processing from old systems to new



Demo #1

http://rmoff.dev/ksldn19_demo-code

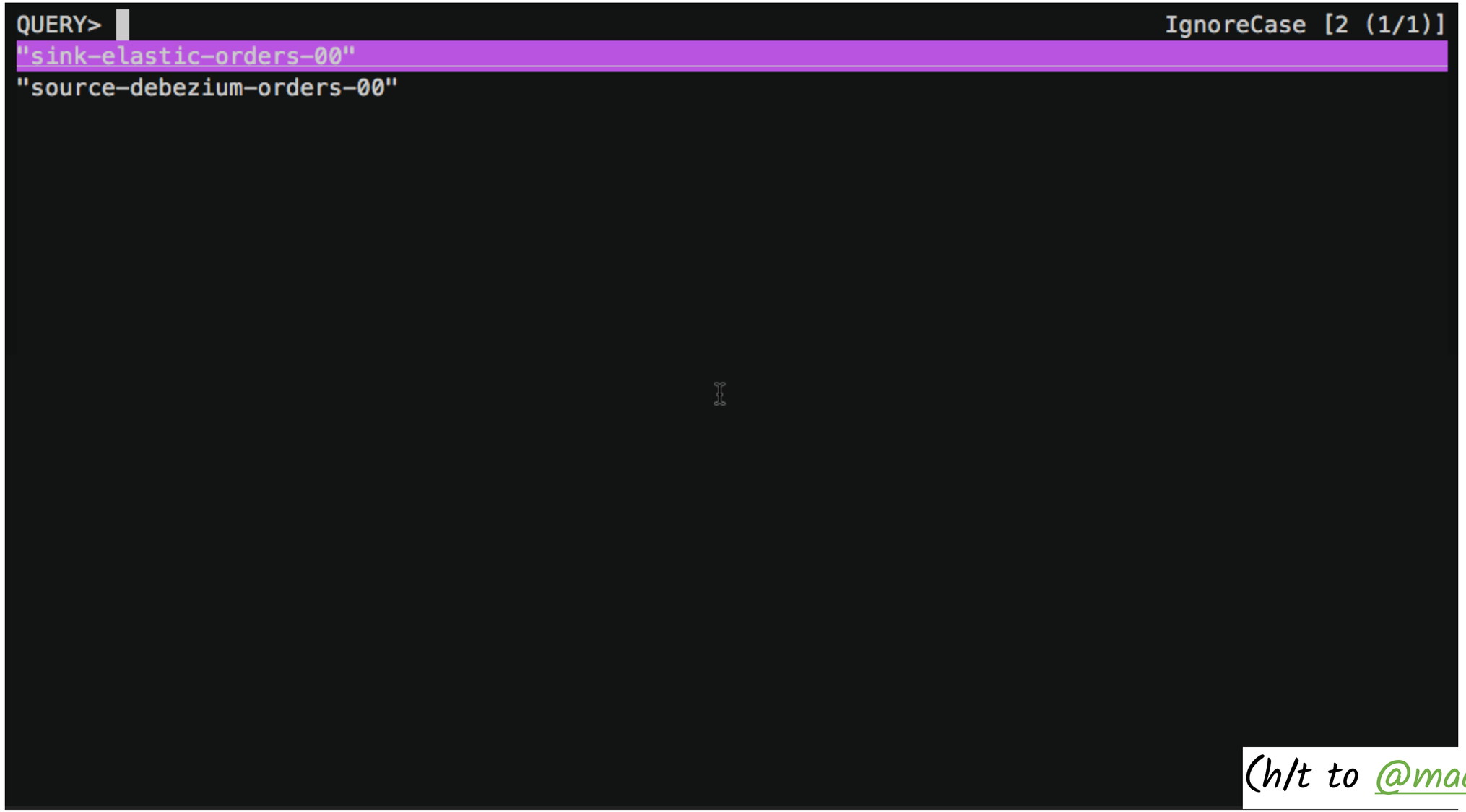
REST API - tricks

```
[~] Robin@asgard02-2.local
$ curl -s "http://localhost:8083/connectors"| \
    jq '.[]'| \
    xargs -I{connector_name} curl -s "http://localhost:8083/connectors/"{connector_name}"/status" | \
    jq -c -M '[.name,.connector.state,.tasks[].state]|join(":|:")' | \
    column -s : -t | \
    sed 's/\"//g' | \
    sort

sink-elastic-orders-00      |  RUNNING  |  RUNNING
sink-elastic-orders-01      |  RUNNING  |  RUNNING
source-debezium-orders-00   |  RUNNING  |  RUNNING
[~] Robin@asgard02-2.local
$
```

http://go.rmoff.net/connector-status

REST API - tricks



(h/t to @madewithtea)

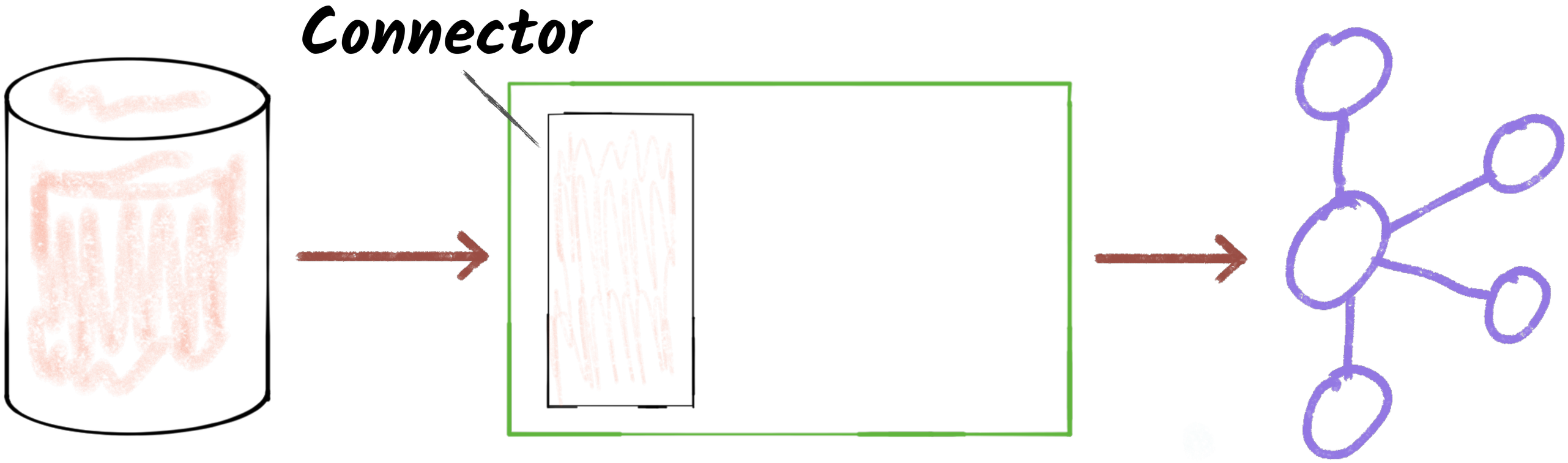
http://go.rmoff.net/selectively-delete-connectors

Configuring Kafka Connect

Inside the API - connectors, transforms, converters

Kafka Connect basics

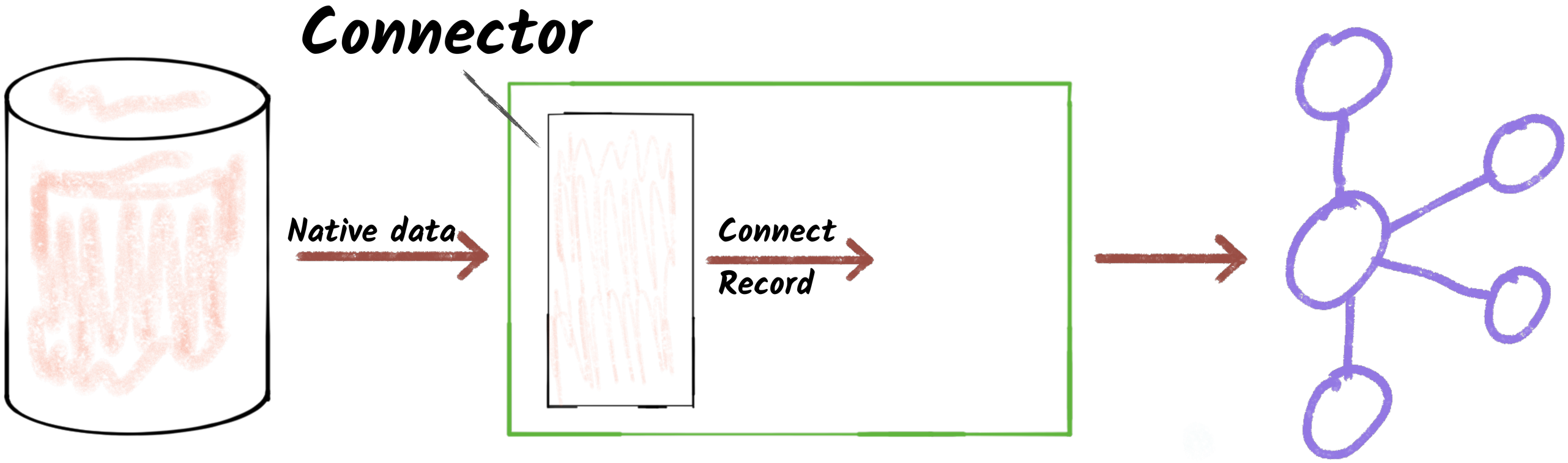
Connectors



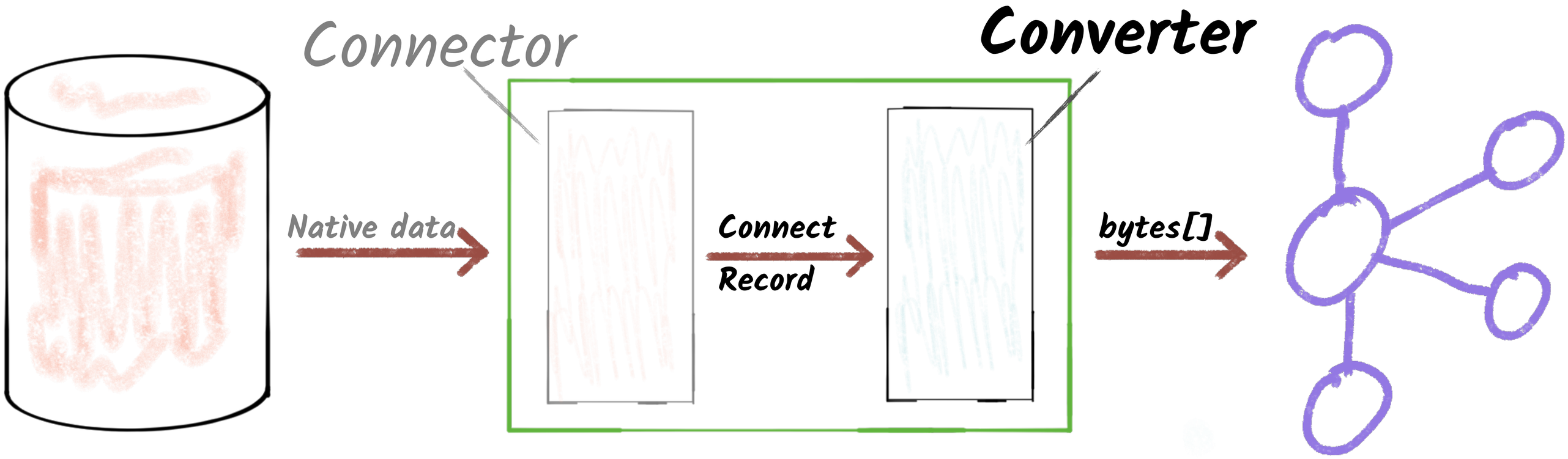
Connectors

```
"config": {
  [...]
  "connector.class": "io.confluent.connect.jdbc.JdbcSinkConnector",
  "connection.url": "jdbc:postgresql://postgres:5432/",
  "topics": "asgard.demo.orders",
}
```

Connectors


Connector
Native data
Connect
Record
Source
Kafka Connect
Kafka

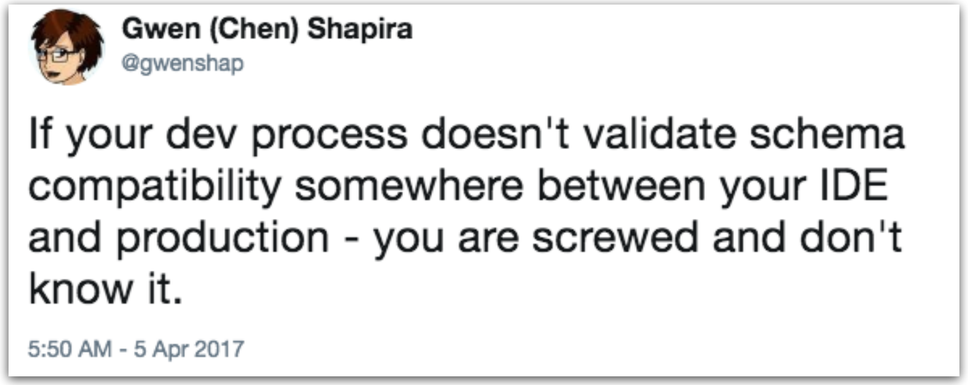
Converters


Connector
Converter
Native data
Connect
Record
bytes[]
Source
Kafka Connect
Kafka

Serialisation & Schemas

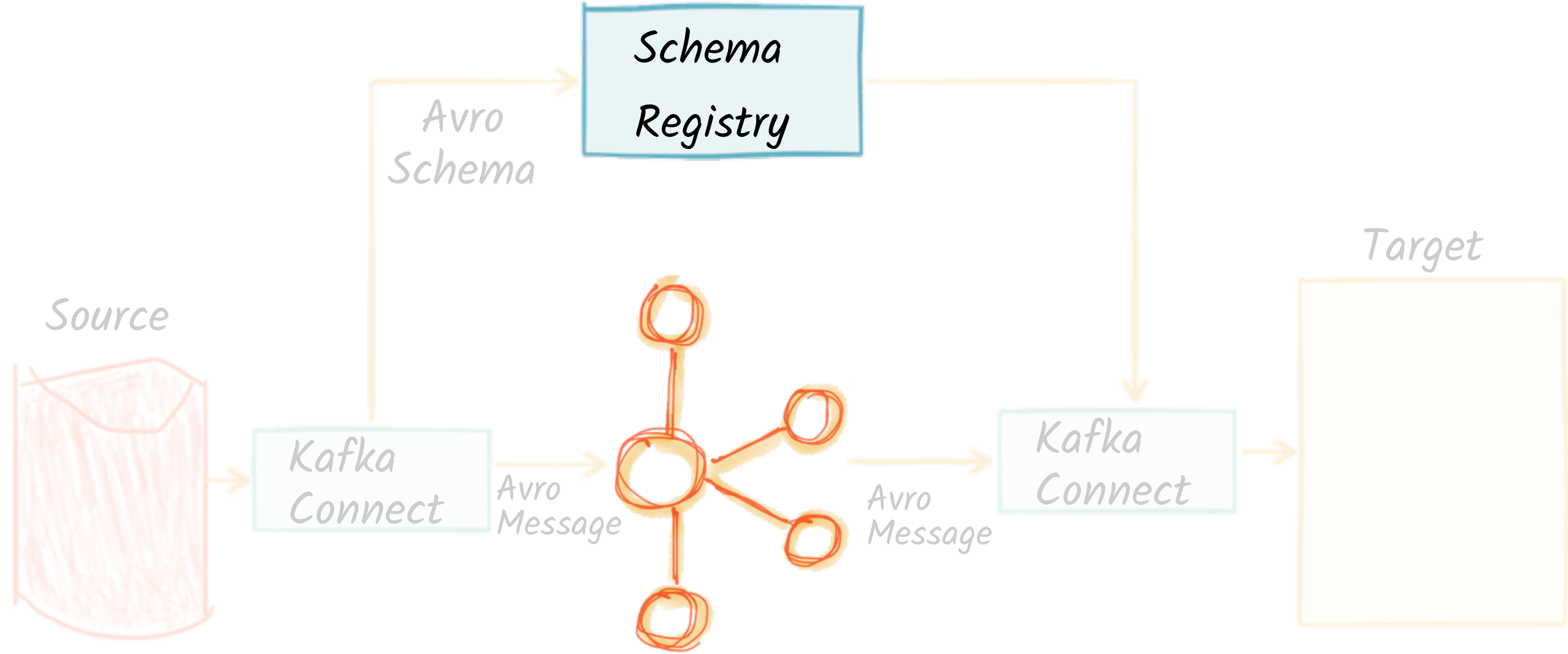
Avro Protobuf JSON CSV

-> Confluent Schema Registry



https://qconnewyork.com/system/files/presentation-slides/qcon_17_-_schemas_and_apis.pdf

The Confluent Schema Registry



Converters

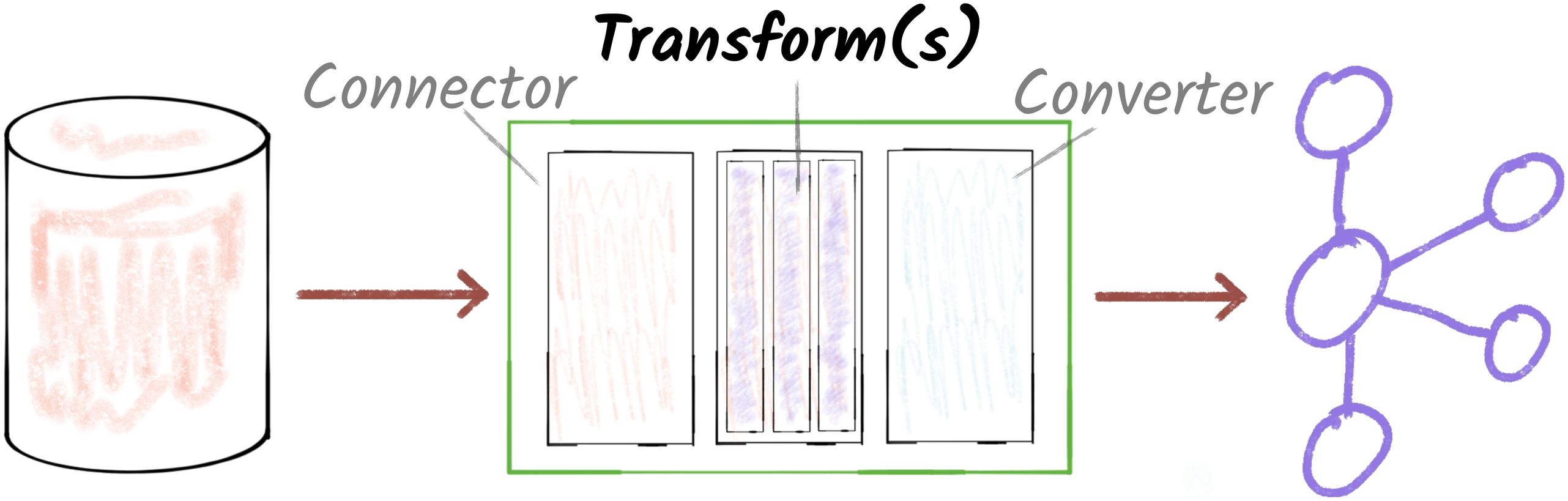
```
key.converter=io.confluent.connect.avro.AvroConverter
key.converter.schema.registry.url=http://localhost:8081
value.converter=io.confluent.connect.avro.AvroConverter
value.converter.schema.registry.url=http://localhost:8081
```

Set as a global default per-worker; optionally can be overriden per-connector

What about internal converters?

```
value.converter=org.apache.kafka.connect.json.JsonConverter
internal.value.converter=org.apache.kafka.connect.json.JsonConverter
key.internal.value.converter=org.apache.kafka.connect.json.JsonConverter
value.internal.value.converter=org.apache.kafka.connect.json.JsonConverter
key.internal.value.converter.bork.bork.bork=org.apache.kafka.connect.json.JsonConverter
key.internal.value.please.just.work.converter=org.apache.kafka.connect.json.JsonConverter
```

Single Message Transforms



Single Message Transforms

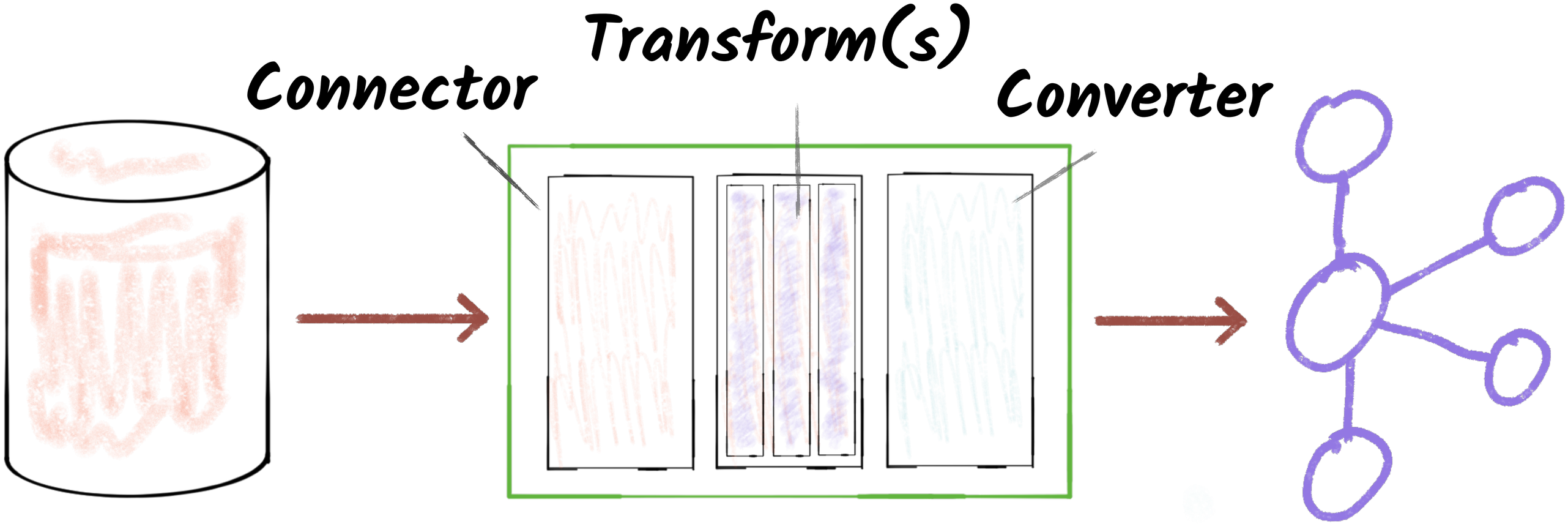
```
"config": {
  [...]
  "transforms": "addDateToTopic,labelFooBar",
  "transforms.addDateToTopic.type": "org.apache.kafka.connect.transforms.TimestampRouter",
  "transforms.addDateToTopic.topic.format": "${topic}-${timestamp}",
  "transforms.addDateToTopic.timestamp.format": "YYYYMM",
  "transforms.labelFooBar.type": "org.apache.kafka.connect.transforms.ReplaceField$Value",
  "transforms.labelFooBar.renames": "delivery_address:shipping_address",
}
```

Do these transforms

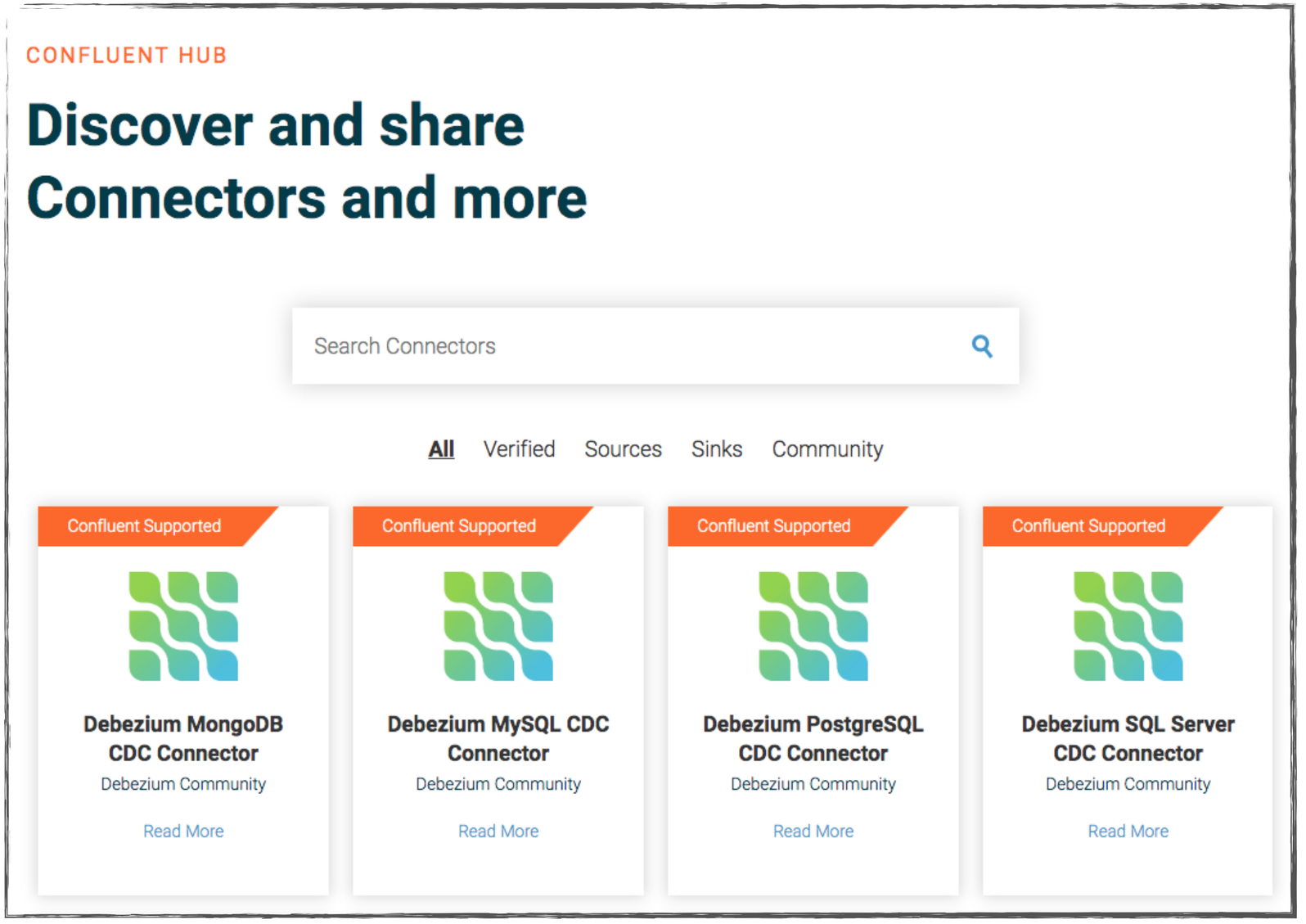
Transforms config

Config per transform

Extensible



Confluent Hub

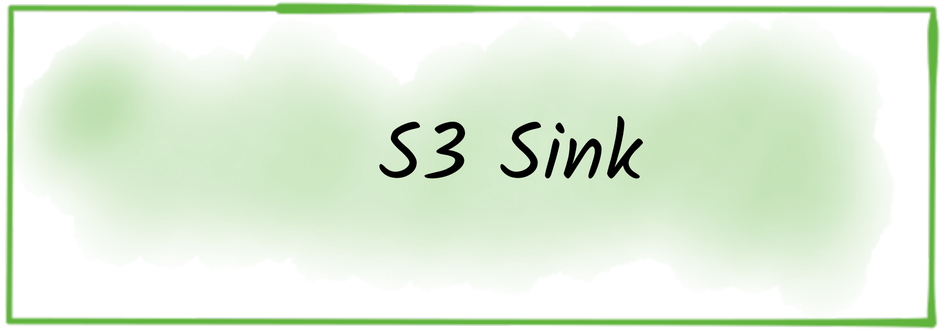


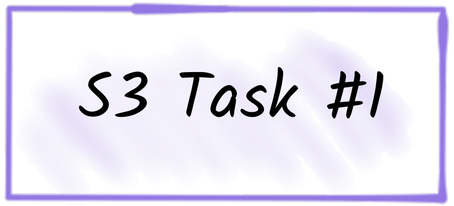
hub.confluent.io

Deploying Kafka Connect

Connectors, Tasks, and Workers

Connectors and Tasks





Connectors and Tasks

JDBC Source

S3 Sink

S3 Task #1

JDBC Task #1

Connectors and Tasks

JDBC Source

S3 Sink

S3 Task #1

JDBC Task #1

JDBC Task #2

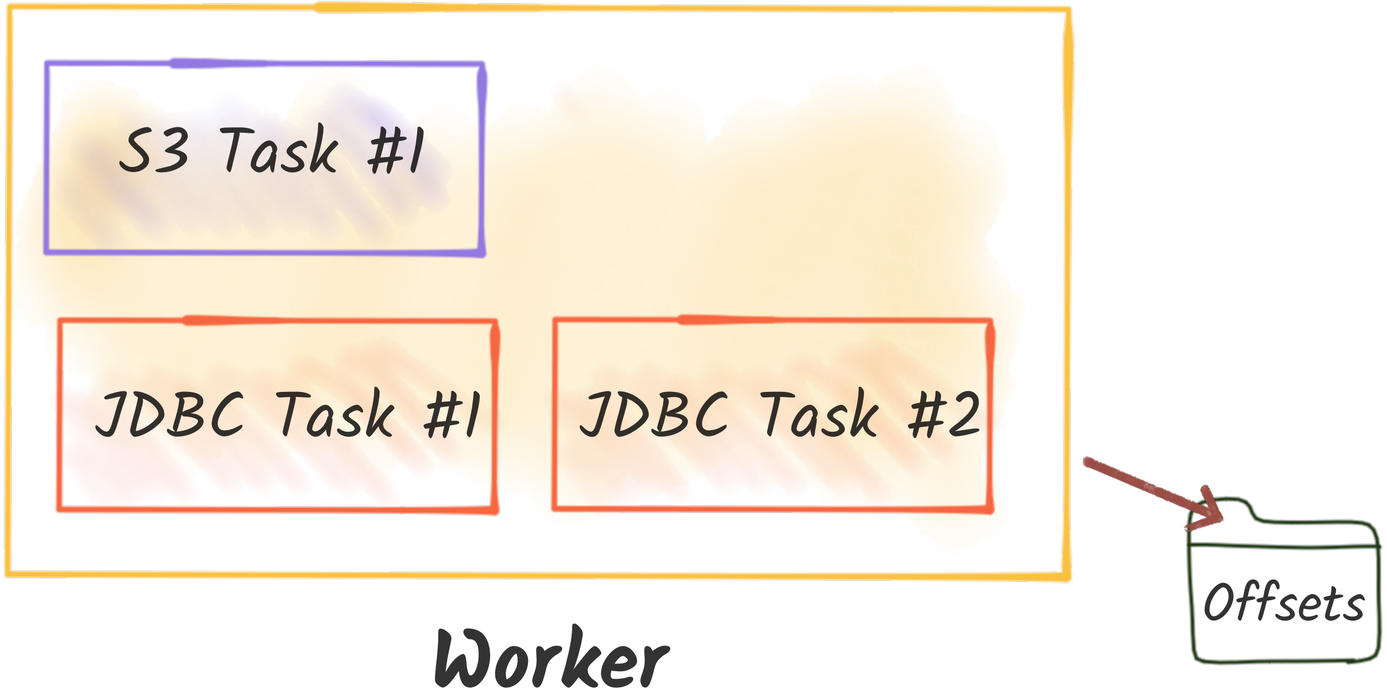
Tasks and Workers

JDBC Source

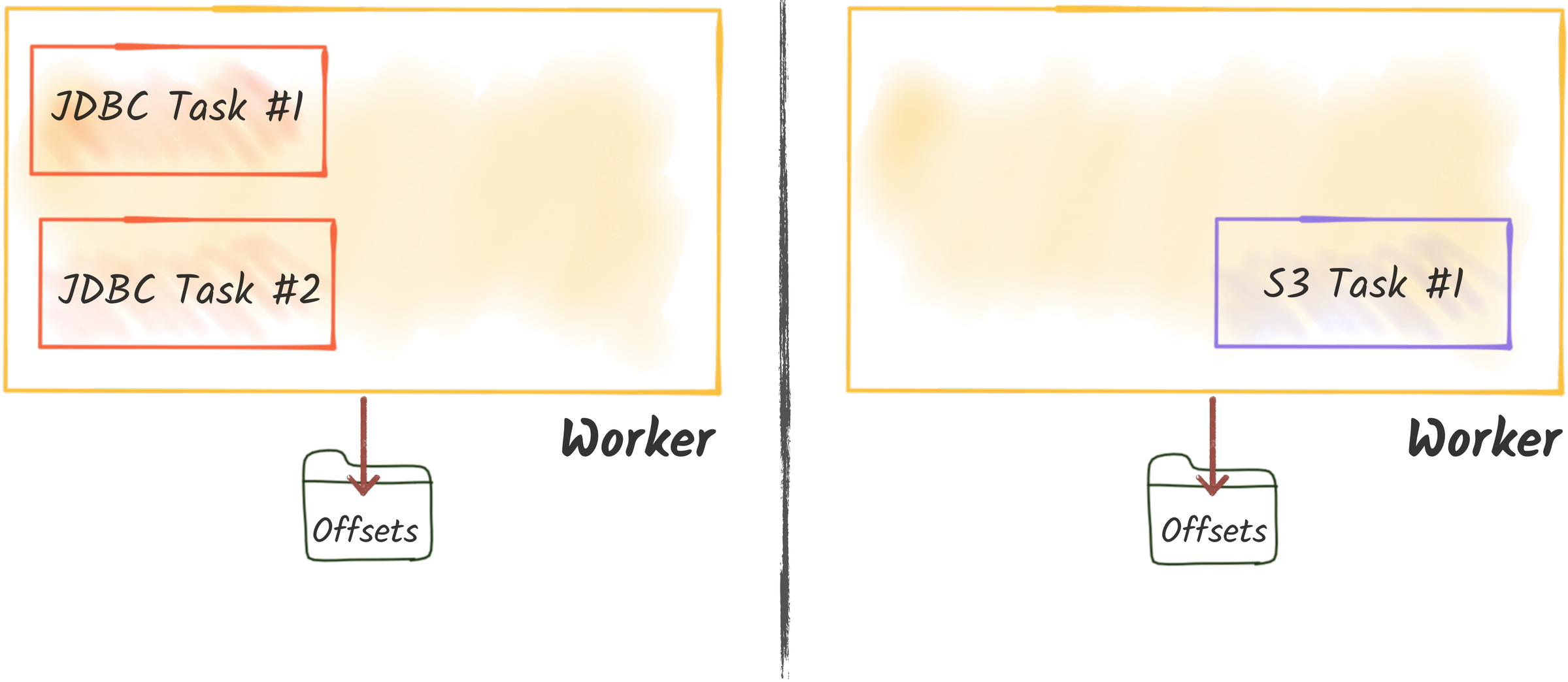
S3 Sink

Worker

Kafka Connect Standalone Worker

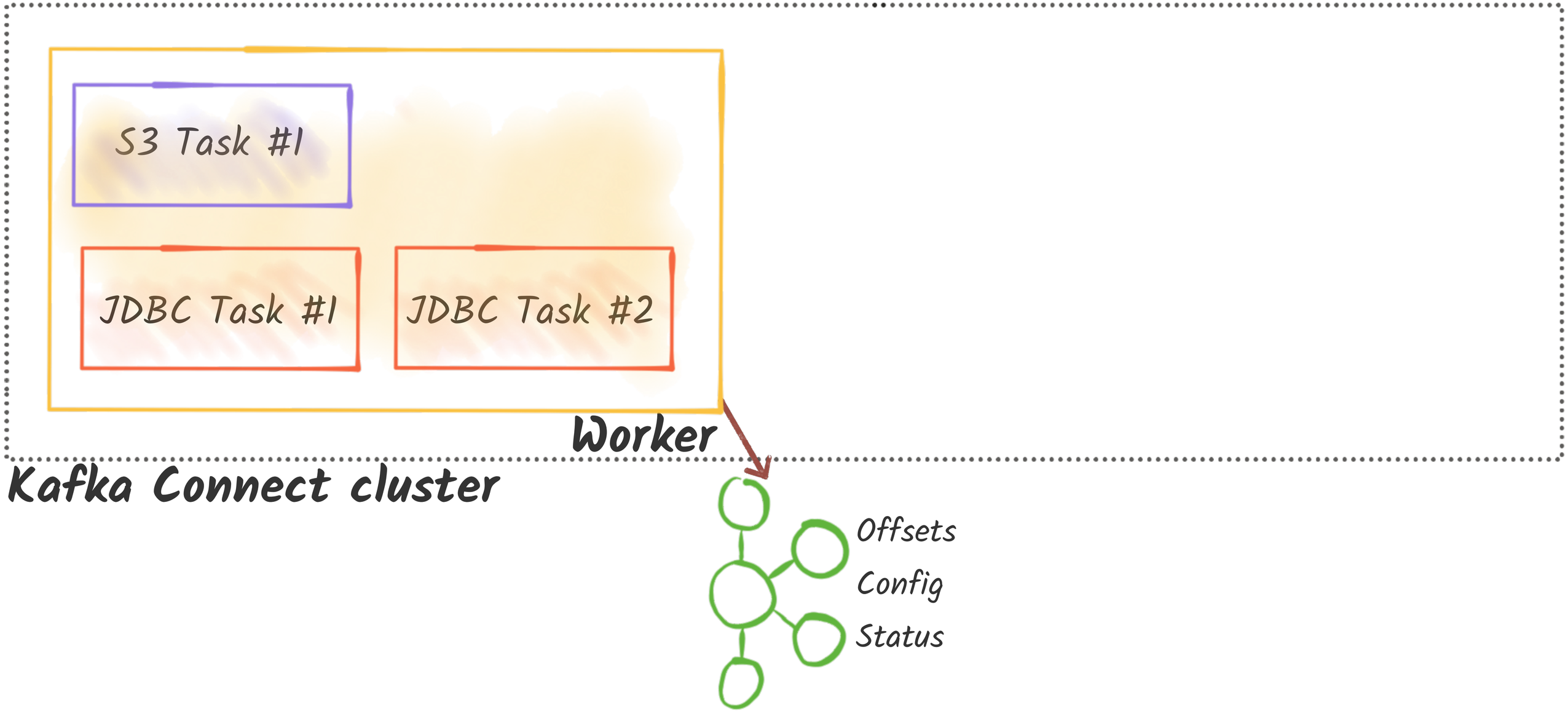


"Scaling" the Standalone Worker



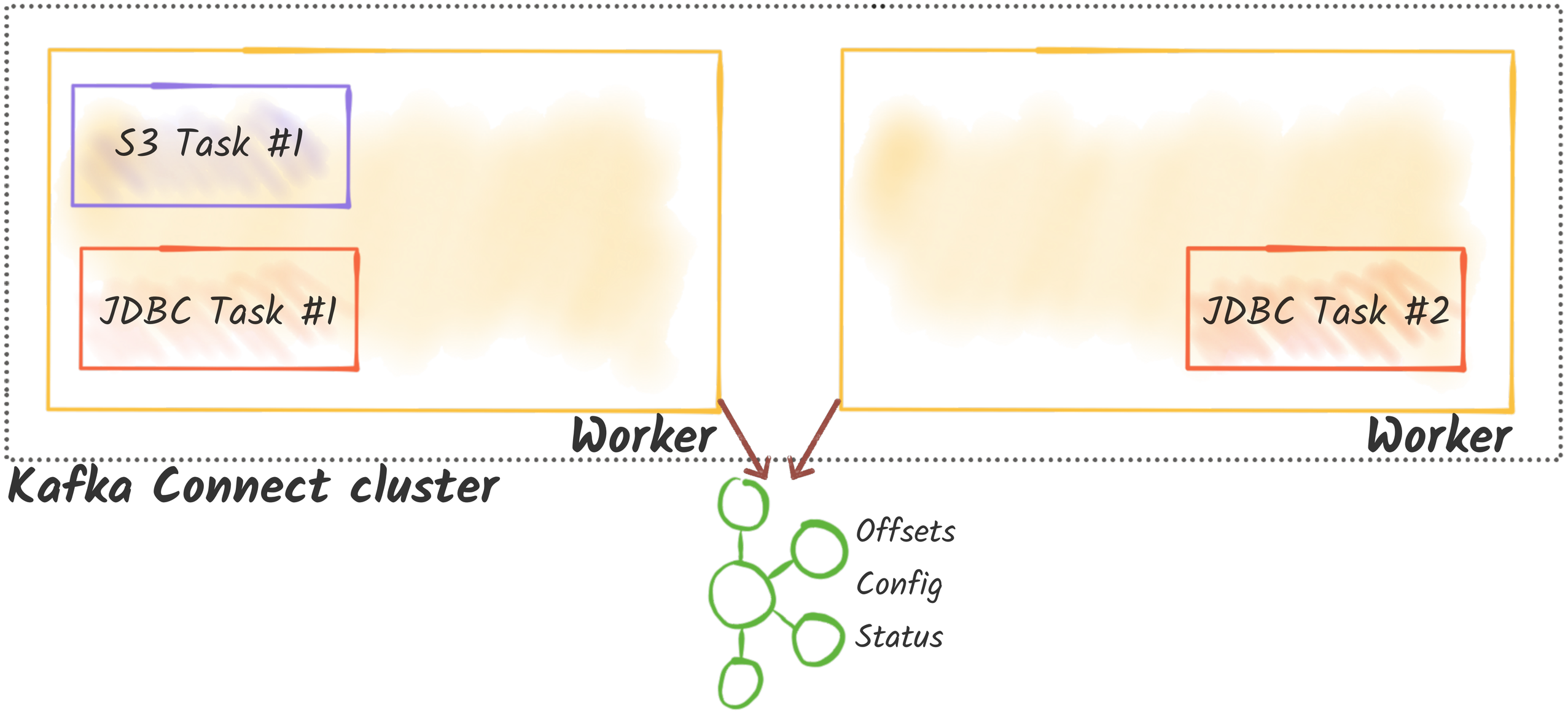
Fault-tolerant? **Nope.**

Kafka Connect Distributed Worker



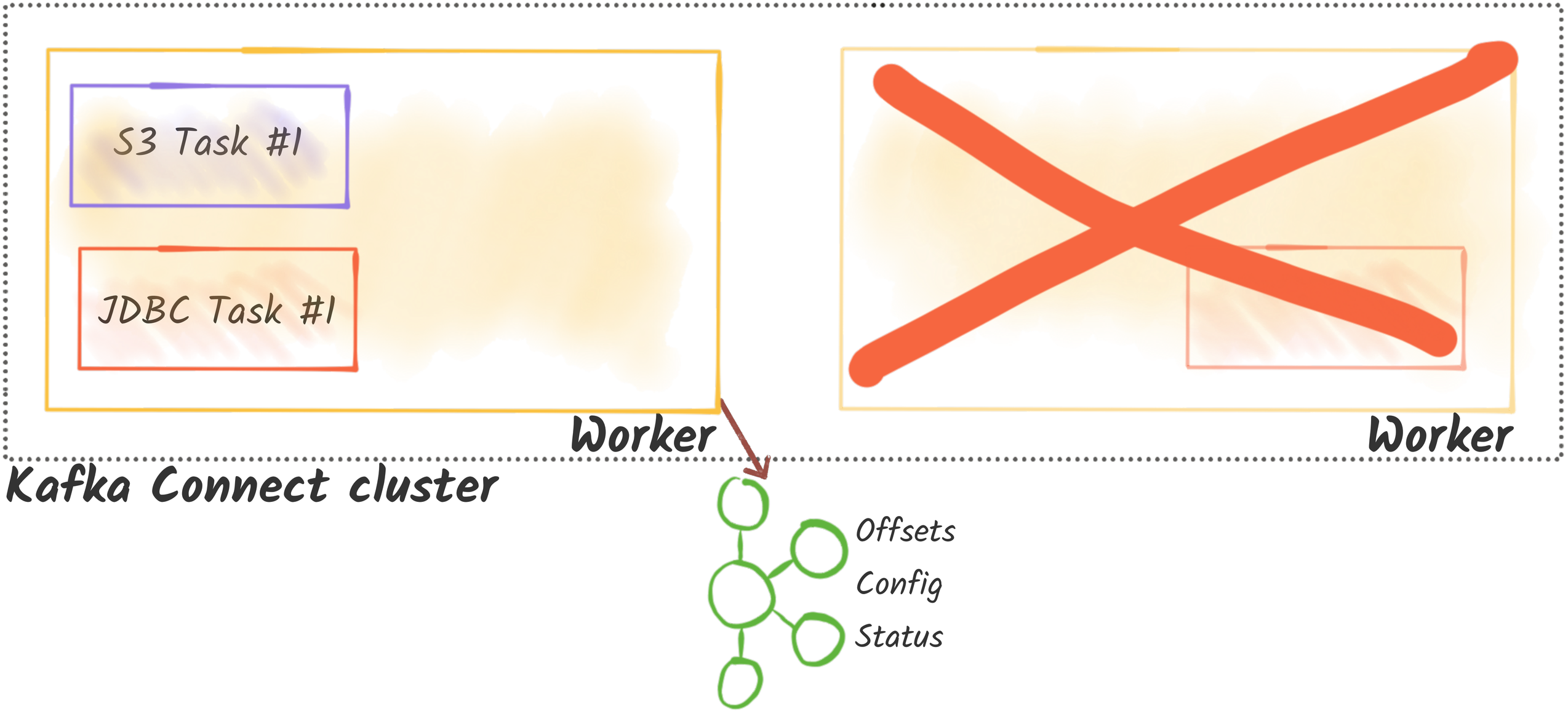
Fault-tolerant? **Yeah!**

Scaling the Distributed Worker

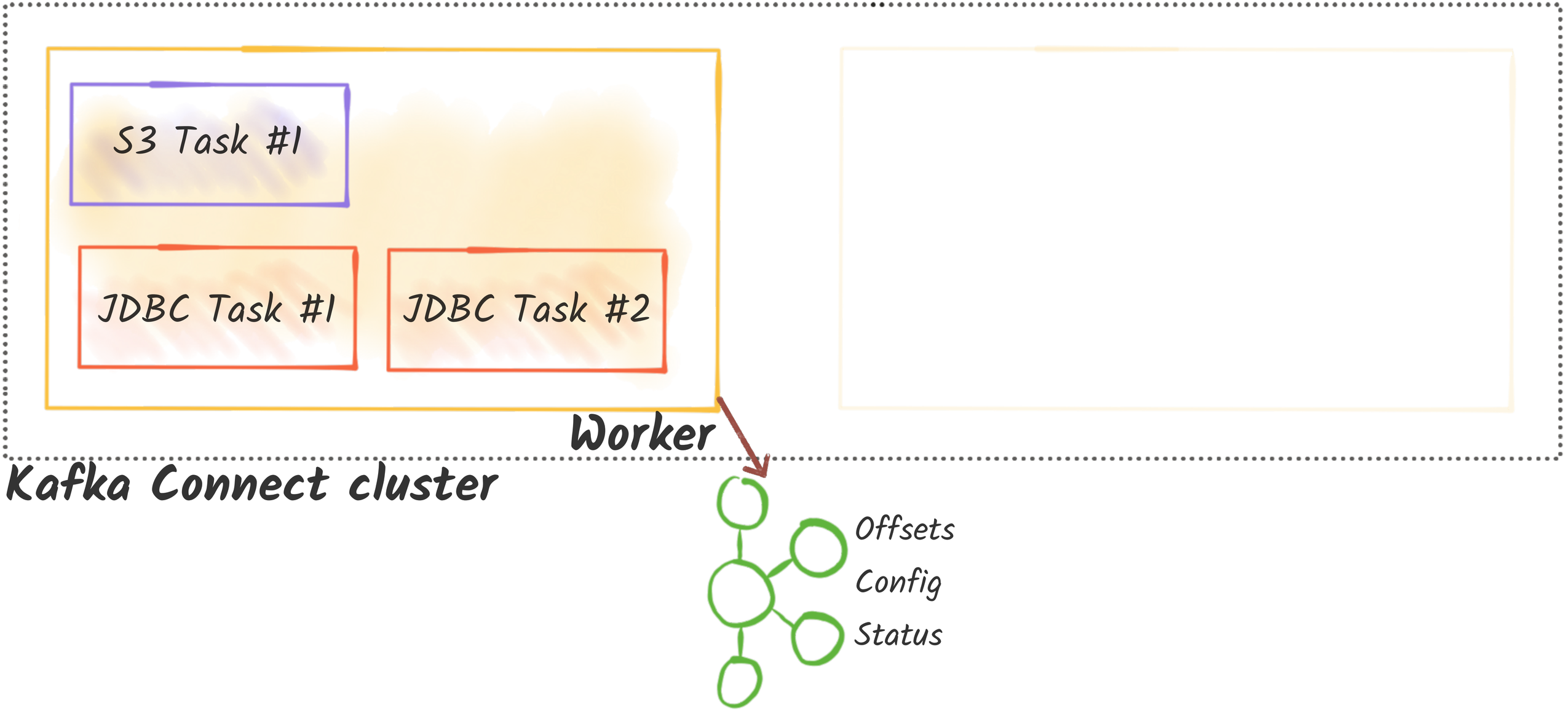


Fault-tolerant? **Yeah!**

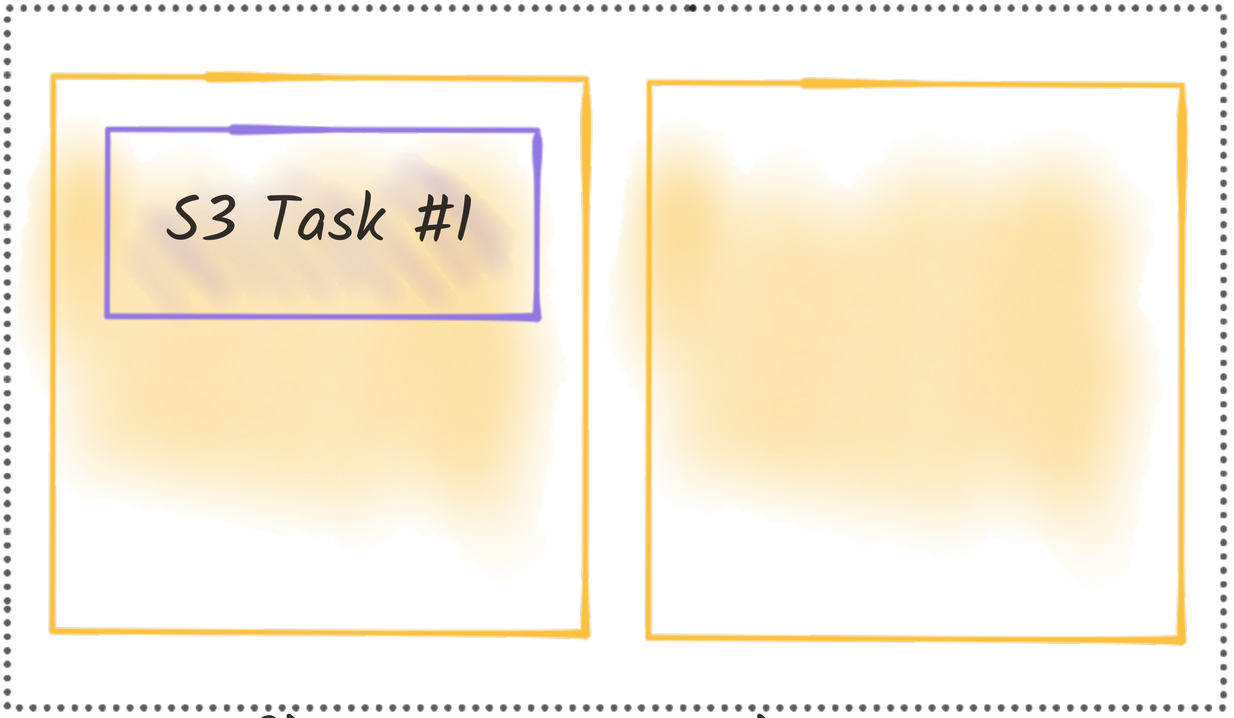
Distributed Worker - fault tolerance



Distributed Worker - fault tolerance



Multiple Distributed Clusters

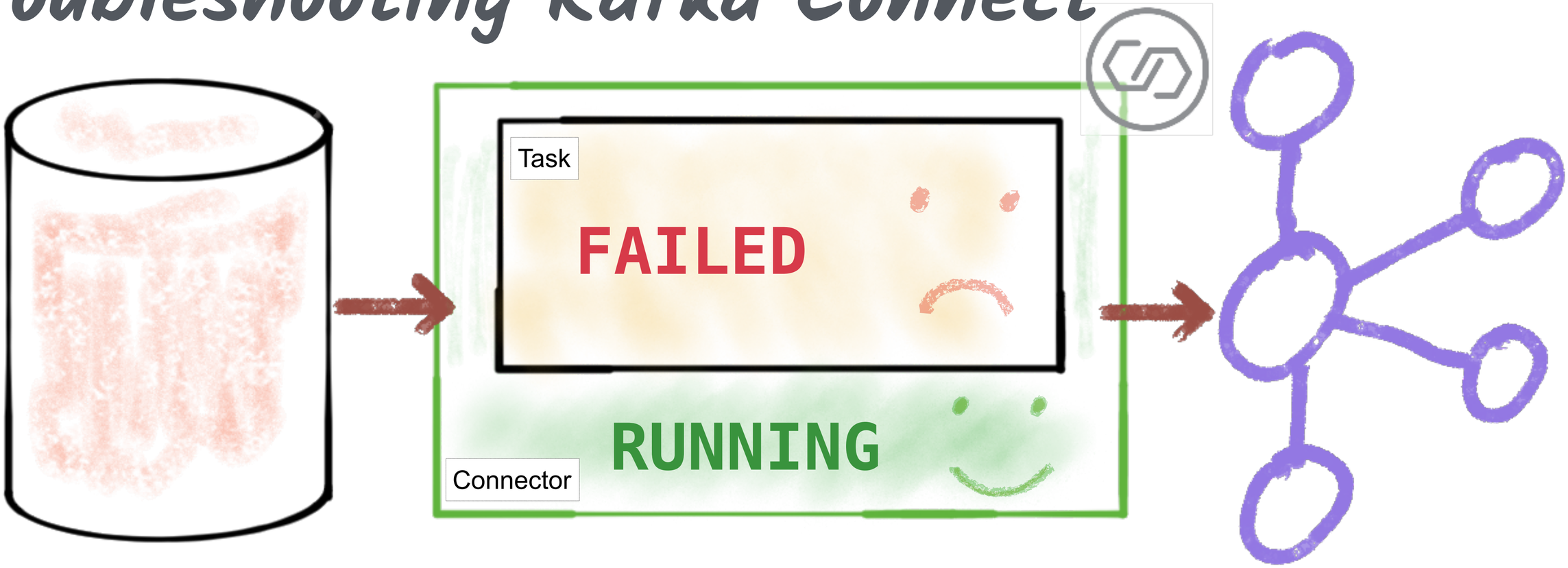


Kafka Connect cluster #1

Kafka Connect cluster #2

Troubleshooting Kafka Connect

Troubleshooting Kafka Connect




```
$ curl -s "http://localhost:8083/connectors/source-debezium-orders/status" | \
  jq '.connector.state'
"RUNNING"
$ curl -s "http://localhost:8083/connectors/source-debezium-orders/status" | \
  jq '.tasks[0].state'
"FAILED"
```

http://go.rmoff.net/connector-status

Troubleshooting Kafka Connect

```
curl -s "http://localhost:8083/connectors/source-debezium-orders-00/status"
| jq '.tasks[0].trace'
```

```
"org.apache.kafka.connect.errors.ConnectException\n\tat
io.debezium.connector.mysql.AbstractReader.wrap(AbstractReader.java:230)\n\tat
io.debezium.connector.mysql.AbstractReader.failed(AbstractReader.java:197)\n\tat
io.debezium.connector.mysql.BinlogReader$ReaderThreadLifecycleListener.onCommunicationFailure(BinlogReader.java:
1018)\n\tat com.github.shyiko.mysql.binlog.BinaryLogClient.listenForEventPackets(BinaryLogClient.java:950)\n\tat
com.github.shyiko.mysql.binlog.BinaryLogClient.connect(BinaryLogClient.java:580)\n\tat
com.github.shyiko.mysql.binlog.BinaryLogClient$7.run(BinaryLogClient.java:825)\n\tat java.lang.Thread.run(Thread.java:
748)\nCaused by: java.io.EOFException\n\tat
com.github.shyiko.mysql.binlog.io.ByteArrayInputStream.read(ByteArrayInputStream.java:190)\n\tat
com.github.shyiko.mysql.binlog.io.ByteArrayInputStream.readInteger(ByteArrayInputStream.java:46)\n\tat
com.github.shyiko.mysql.binlog.event.deserialization.EventHeaderV4Deserializer.deserialize(EventHeaderV4Deserializer.java
:35)\n\tat
com.github.shyiko.mysql.binlog.event.deserialization.EventHeaderV4Deserializer.deserialize(EventHeaderV4Deserializer.java
:27)\n\tat com.github.shyiko.mysql.binlog.event.deserialization.EventDeserializer.nextEvent(EventDeserializer.java:
212)\n\tat io.debezium.connector.mysql.BinlogReader$1.nextEvent(BinlogReader.java:224)\n\tat
com.github.shyiko.mysql.binlog.BinaryLogClient.listenForEventPackets(BinaryLogClient.java:922)\n\t ... 3 more\n"
```

The log is the source of truth

```
$ confluent log connect

$ docker-compose logs kafka-connect

$ cat /var/log/kafka/connect.log
```

Kafka Connect

```
ache.kafka.connect.runtime.WorkerSourceTask)
[2019-05-07 14:39:13,115] INFO WorkerSourceTask{id=source-debezium-orders-00-0} Finished commitOffsets successfully in 28 ms (org.apache.
kafka.connect.runtime.WorkerSourceTask)
[2019-05-07 14:39:13,116] ERROR WorkerSourceTask{id=source-debezium-orders-00-0} Task threw an uncaught and unrecoverable exception (org.
apache.kafka.connect.runtime.WorkerTask)
org.apache.kafka.connect.errors.ConnectException
   at io.debezium.connector.mysql.AbstractReader.wrap(AbstractReader.java:230)
   at io.debezium.connector.mysql.AbstractReader.failed(AbstractReader.java:197)
   at io.debezium.connector.mysql.BinlogReader$ReaderThreadLifecycleListener.onCommunicationFailure(BinlogReader.java:1018)
   at com.github.shyiko.mysql.binlog.BinaryLogClient.listenForEventPackets(BinaryLogClient.java:950)
   at com.github.shyiko.mysql.binlog.BinaryLogClient.connect(BinaryLogClient.java:580)
   at com.github.shyiko.mysql.binlog.BinaryLogClient$7.run(BinaryLogClient.java:825)
   at java.lang.Thread.run(Thread.java:748)
Caused by: java.io.EOFException
   at com.github.shyiko.mysql.binlog.io.ByteArrayInputStream.read(ByteArrayInputStream.java:190)
   at com.github.shyiko.mysql.binlog.io.ByteArrayInputStream.readInteger(ByteArrayInputStream.java:46)
   at com.github.shyiko.mysql.binlog.event.deserialization.EventHeaderV4Deserializer.deserialize(EventHeaderV4Deserializer.java:35)
   at com.github.shyiko.mysql.binlog.event.deserialization.EventHeaderV4Deserializer.deserialize(EventHeaderV4Deserializer.java:27)
   at com.github.shyiko.mysql.binlog.event.deserialization.EventDeserializer.nextEvent(EventDeserializer.java:212)
   at io.debezium.connector.mysql.BinlogReader$1.nextEvent(BinlogReader.java:224)
   at com.github.shyiko.mysql.binlog.BinaryLogClient.listenForEventPackets(BinaryLogClient.java:922)
   ... 3 more
[2019-05-07 14:39:13,121] ERROR WorkerSourceTask{id=source-debezium-orders-00-0} Task is being killed and will not recover until manually restarted
 (org.apache.kafka.connect.runtime.WorkerTask)
```

Kafka Connect

"Task is being killed and will not recover until manually restarted"

Symptom not Cause

Kafka Connect

```
ache.kafka.connect.runtime.WorkerSourceTask)
[2019-05-07 14:39:13,115] INFO WorkerSourceTask{id=source-debezium-orders-00-0} Finished commitOffsets successfully in 28 ms (org.apache.
kafka.connect.runtime.WorkerSourceTask)
[2019-05-07 14:39:13,116] ERROR WorkerSourceTask{id=source-debezium-orders-00-0} Task threw an uncaught and unrecoverable exception (org.
apache.kafka.connect.runtime.WorkerTask)
org.apache.kafka.connect.errors.ConnectException
   at io.debezium.connector.mysql.AbstractReader.wrap(AbstractReader.java:230)
   at io.debezium.connector.mysql.AbstractReader.failed(AbstractReader.java:197)
   at io.debezium.connector.mysql.BinlogReader$ReaderThreadLifecycleListener.onCommunicationFailure(BinlogReader.java:1018)
   at com.github.shyiko.mysql.binlog.BinaryLogClient.listenForEventPackets(BinaryLogClient.java:950)
   at com.github.shyiko.mysql.binlog.BinaryLogClient.connect(BinaryLogClient.java:580)
   at com.github.shyiko.mysql.binlog.BinaryLogClient$7.run(BinaryLogClient.java:825)
   at java.lang.Thread.run(Thread.java:748)
Caused by: java.io.EOFException
   at com.github.shyiko.mysql.binlog.io.ByteArrayInputStream.read(ByteArrayInputStream.java:190)
   at com.github.shyiko.mysql.binlog.io.ByteArrayInputStream.readInteger(ByteArrayInputStream.java:46)
   at com.github.shyiko.mysql.binlog.event.deserialization.EventHeaderV4Deserializer.deserialize(EventHeaderV4Deserializer.java:35)
   at com.github.shyiko.mysql.binlog.event.deserialization.EventHeaderV4Deserializer.deserialize(EventHeaderV4Deserializer.java:27)
   at com.github.shyiko.mysql.binlog.event.deserialization.EventDeserializer.nextEvent(EventDeserializer.java:212)
   at io.debezium.connector.mysql.BinlogReader$1.nextEvent(BinlogReader.java:224)
   at com.github.shyiko.mysql.binlog.BinaryLogClient.listenForEventPackets(BinaryLogClient.java:922)
   ... 3 more
[2019-05-07 14:39:13,121] ERROR WorkerSourceTask{id=source-debezium-orders-00-0} Task is being killed and will not recover until manually restarted
 (org.apache.kafka.connect.runtime.WorkerTask)
```

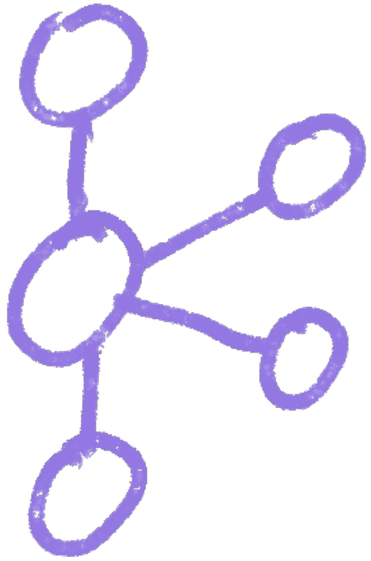
Common errors

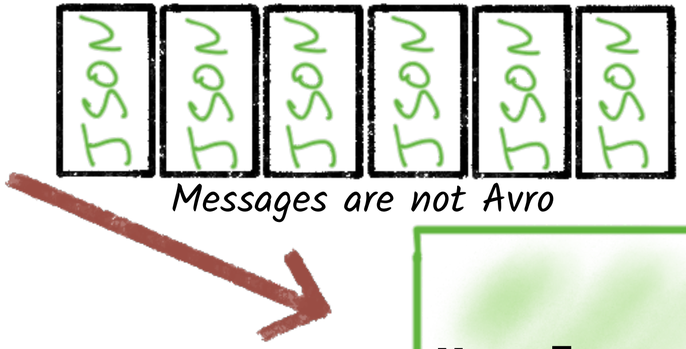
org.apache.kafka.common.errors.SerializationException:
Unknown magic byte!

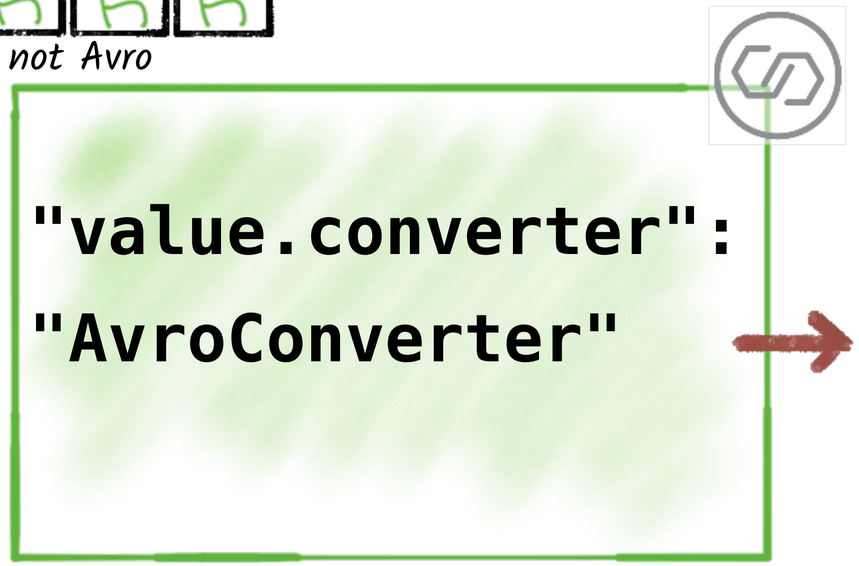
Mismatched converters

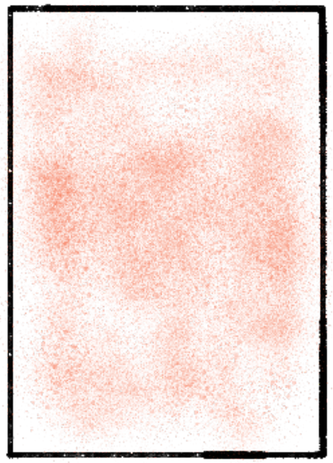
org.apache.kafka.common.errors.SerializationException:

Unknown magic byte!







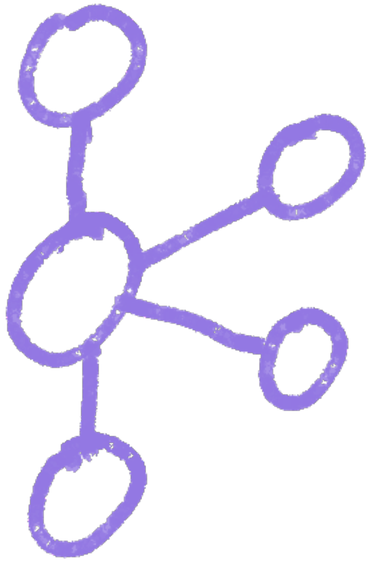


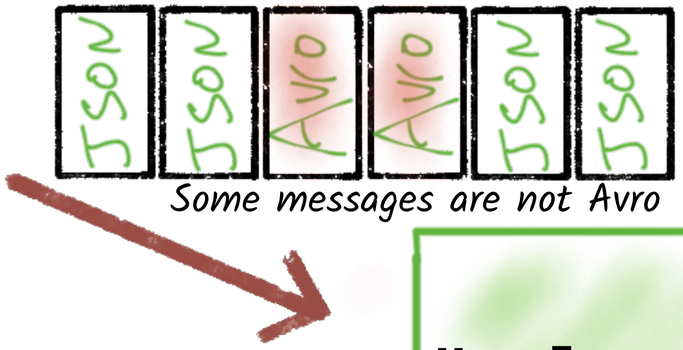
ⓘ Use the correct Converter for the source data

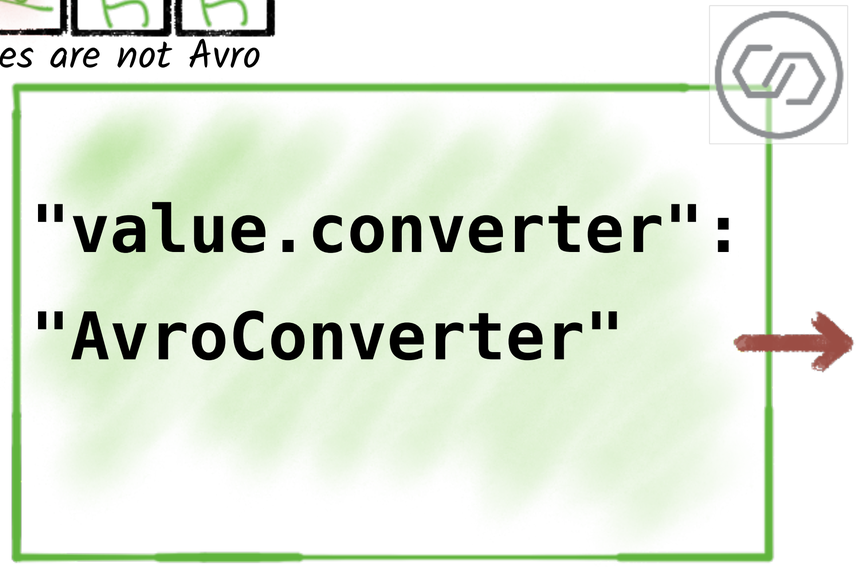
Mixed serialisation methods

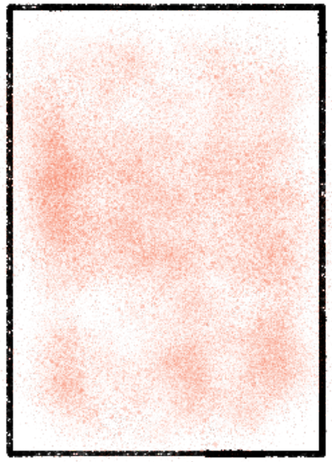
org.apache.kafka.common.errors.SerializationException:

Unknown magic byte!









ⓘ Use error handling to deal with bad messages

Error Handling and DLQ

Handled

Convert (read/write from Kafka, [de]-serialisation)

Transform

Not Handled

Start (Connections to a data store)

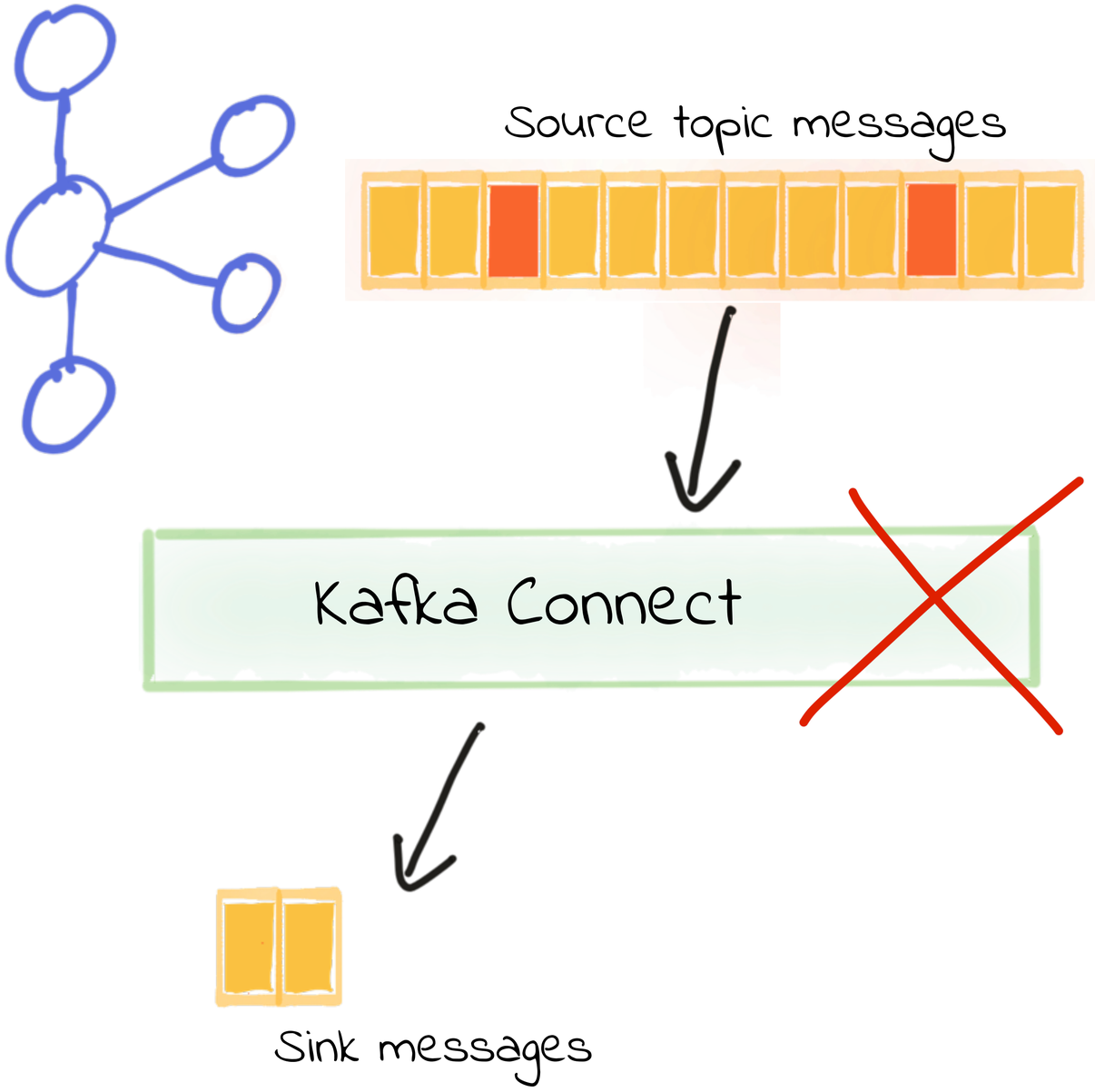
Poll / Put (Read/Write from/to data store)*

* can be retried by Connect

https://cnfl.io/connect-dlq

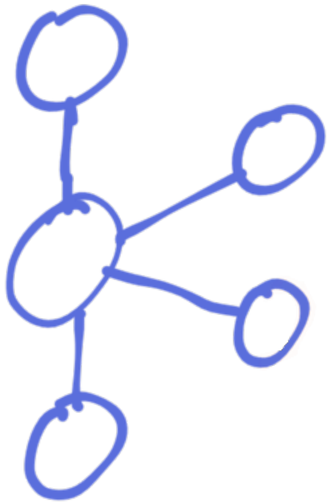
confluent

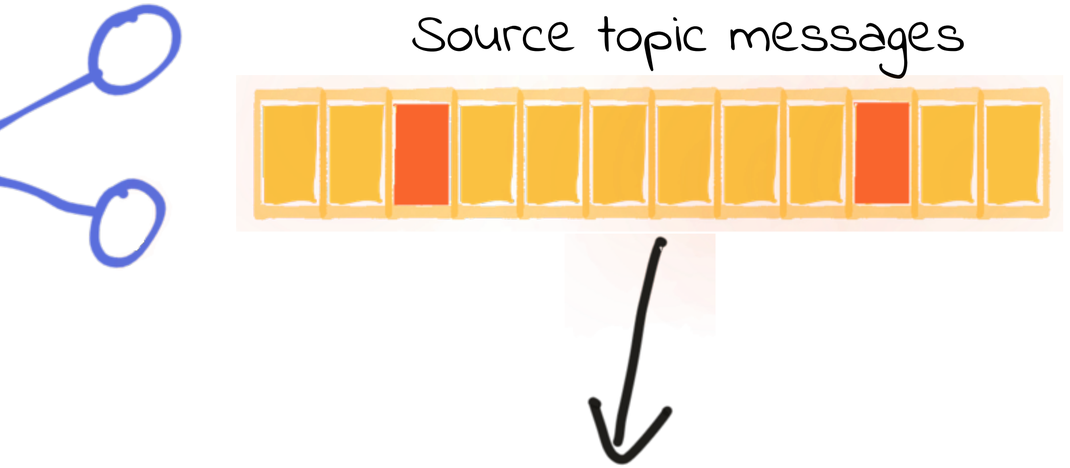
Fail Fast



https://cnfl.io/connect-dlq

YOLO ¯_(ツ)_/¯



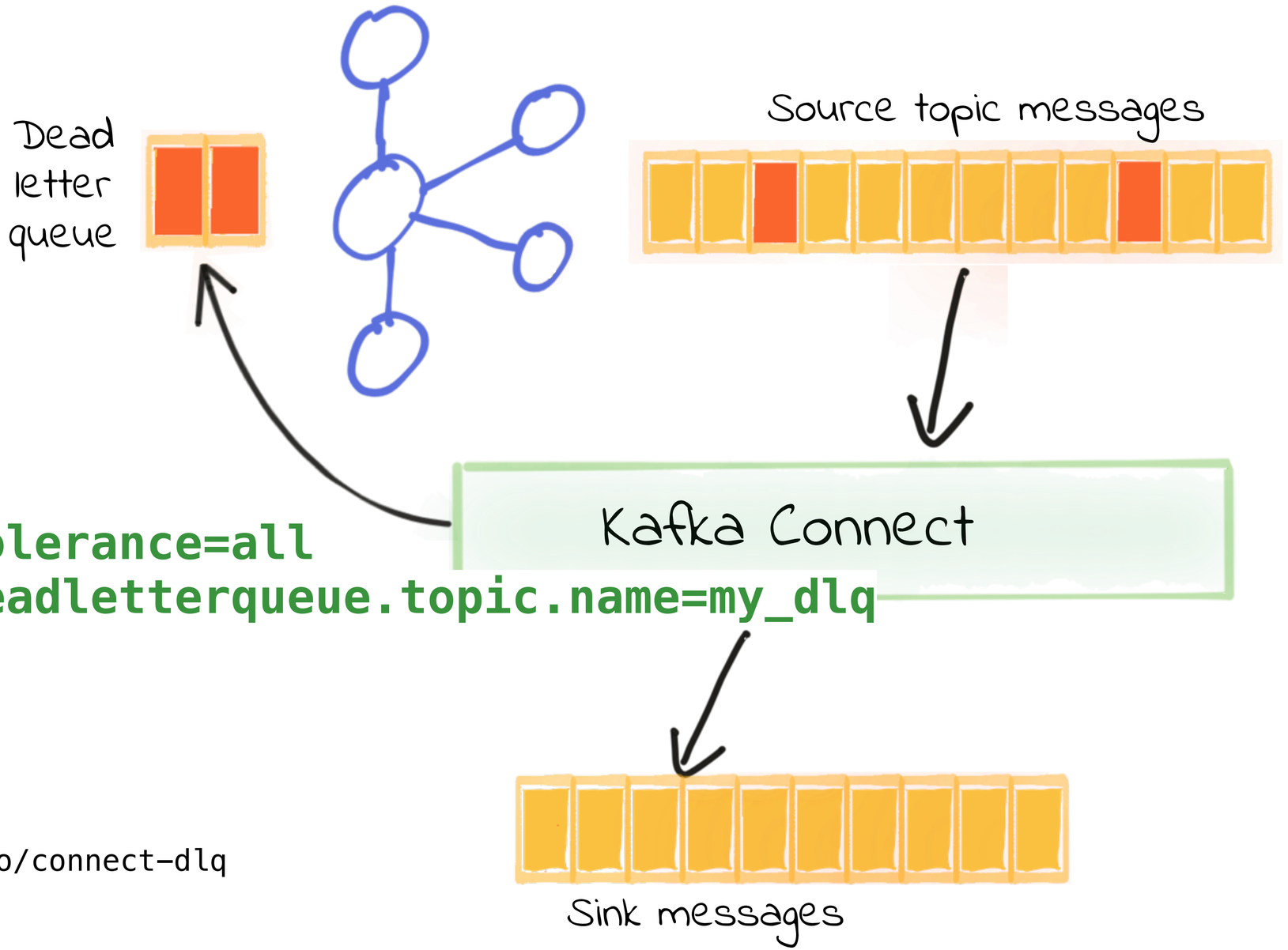



```
errors.tolerance=all
```

https://cnfl.io/connect-dlq

confluent

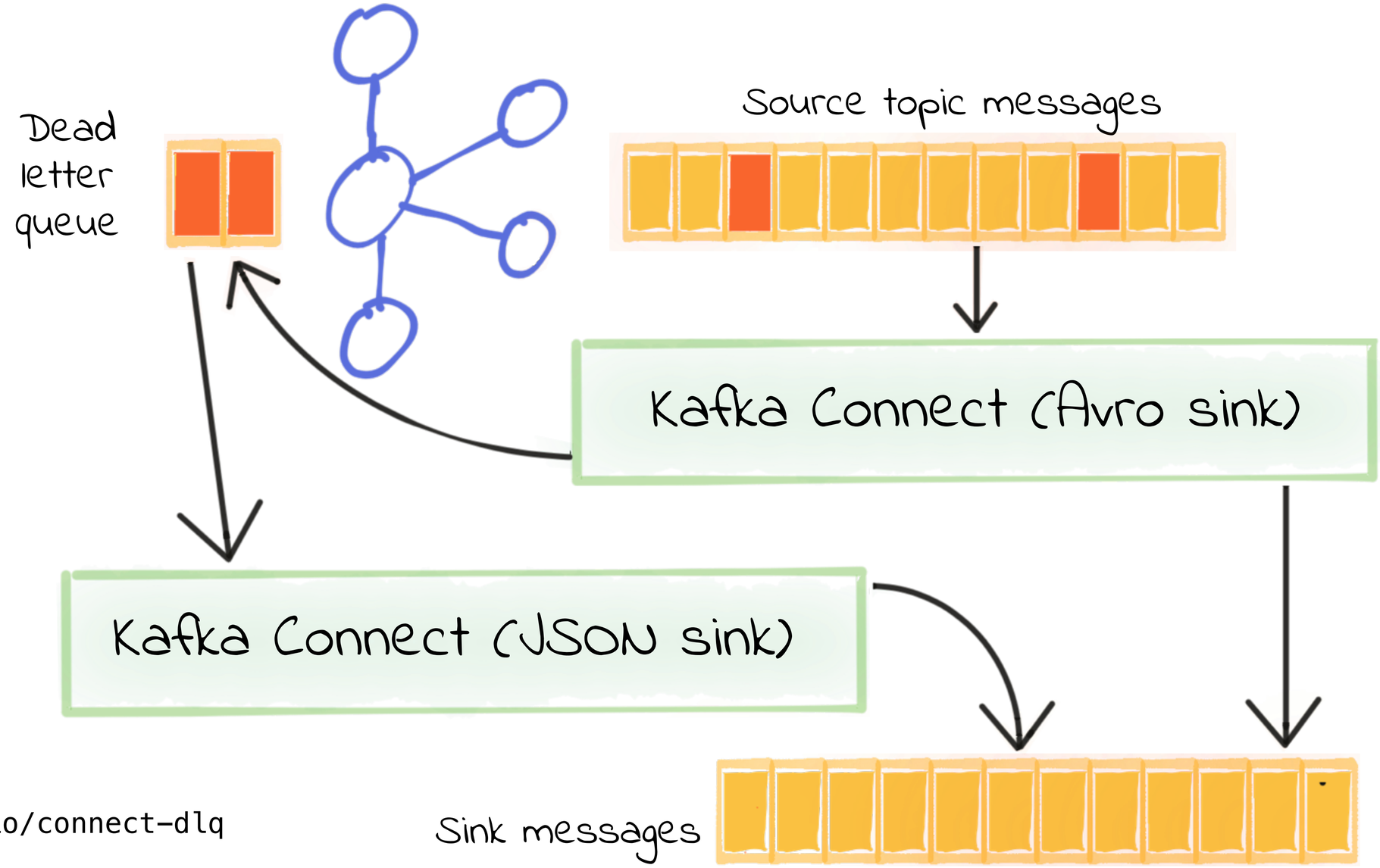
Dead Letter Queue



https://cnfl.io/connect-dlq

confluent

Re-processing the Dead Letter Queue



https://cnfl.io/connect-dlq

No fields found using key and value schemas for table: foo-bar

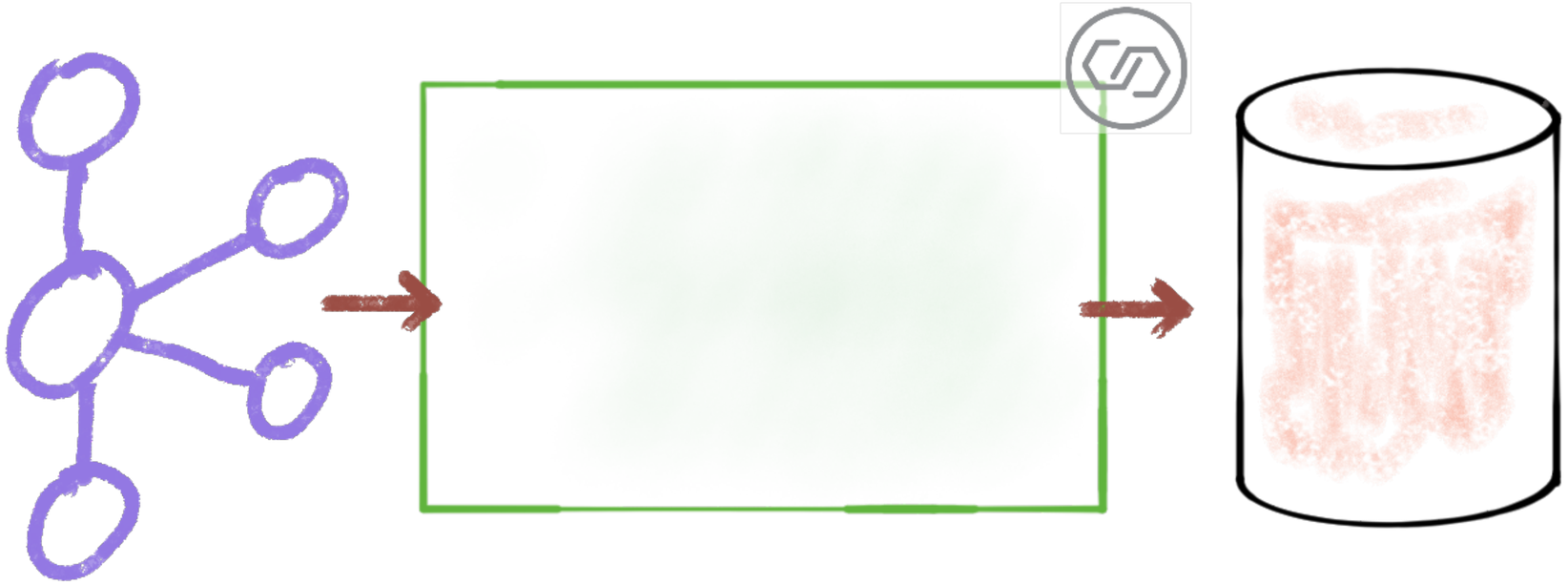
No fields found using key and value schemas for table: foo-bar

JsonDeserializer with schemas.enable requires "schema" and "payload" fields and may not contain additional fields

Schema, where art thou?

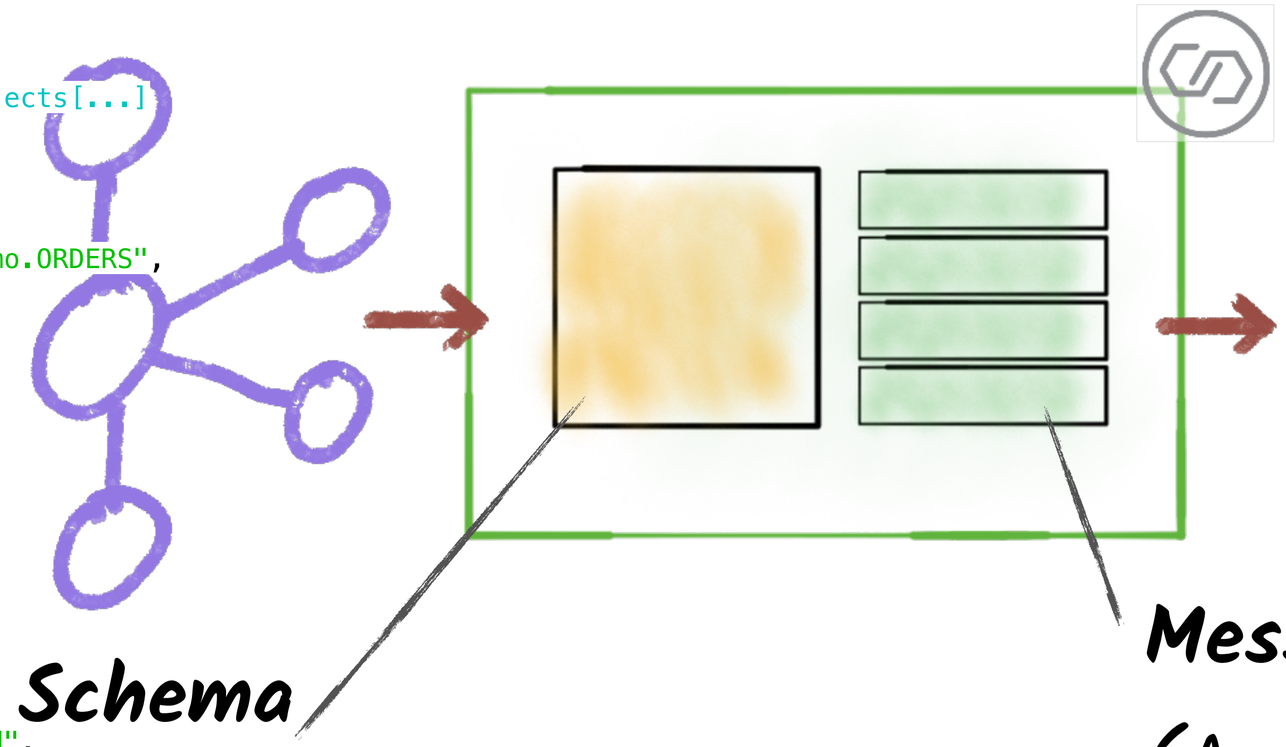
No fields found using key and value schemas for table: foo-bar

JsonDeserializer with schemas.enable requires "schema" and "payload" fields and may not contain additional fields



Schema, where art thou?

```
$ http localhost:8081/subjects[...]
  jq '.schema|fromjson'
{
  "type": "record",
  "name": "Value",
  "namespace": "asgard.demo.ORDERS",
  "fields": [
    {
      "name": "id",
      "type": "int"
    },
    {
      "name": "order_id",
      "type": [
        "null",
        "int"
      ],
      "default": null
    },
    {
      "name": "customer_id",
      "type": [
        "null",
        "int"
      ],
      "default": null
    },
```



Schema

(From Schema Registry)

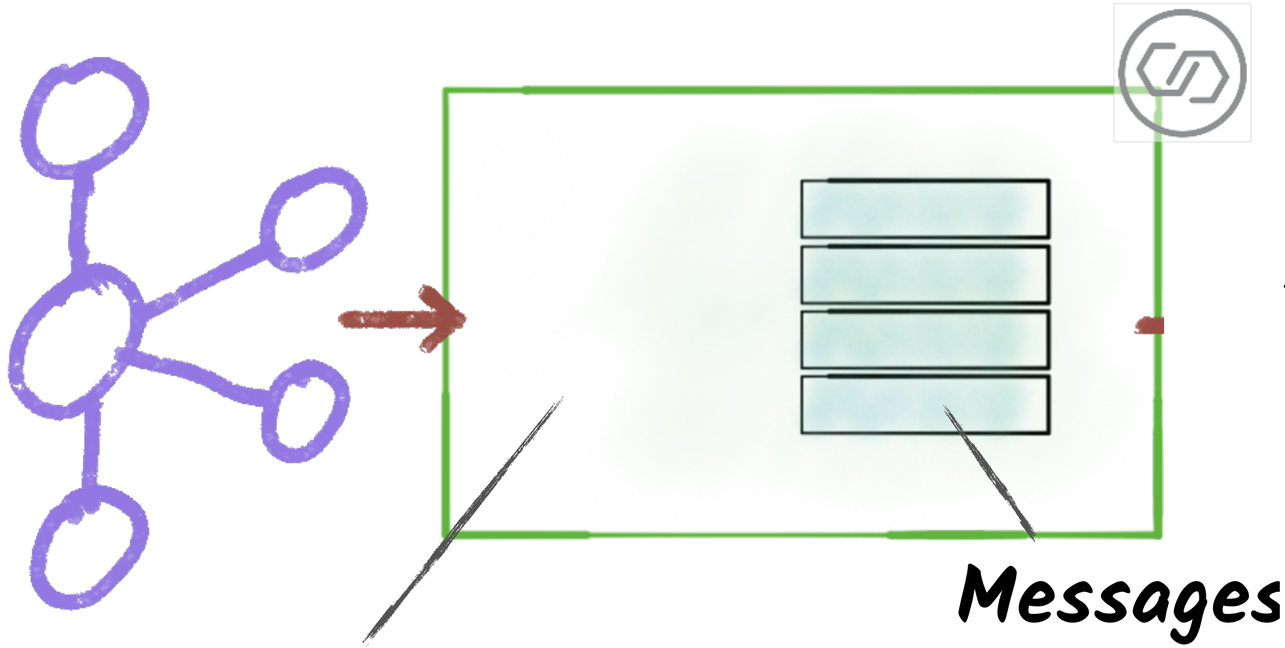
Messages

(Avro)

```
$ kafkacat -b localhost:9092 -C -t mysql-
QF@Land RoverDefender 90SheffieldSwift LE
Parkway(2019-05-09T13:42:28Z(2019-05-09T
```

Schema, where art thou?

No fields found using key and value schemas for table: foo-bar



No schema!

Messages

(JSON)


```
{
  "id": 7,
  "order_id": 7,
  "customer_id": 849,
  "order_total_usd": 98062.21,
  "make": "Pontiac",
  "model": "Aztek",
  "delivery_city": "Leeds",
  "delivery_company": "Price-Zieme
  "delivery_address": "754 Becker
  "CREATE_TS": "2019-05-09T13:42:28
  "UPDATE_TS": "2019-05-09T13:42:28
}
```

Schema, where art thou?

No fields found using key and value schemas for table: foo-bar

-> You need a schema!

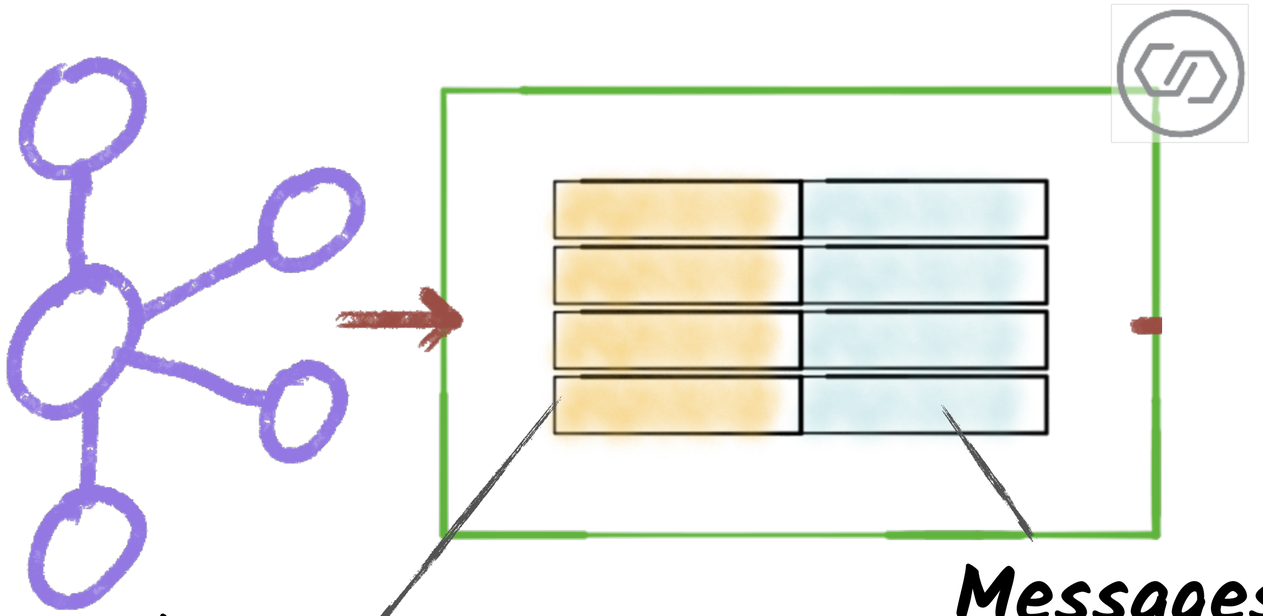
-> Use Avro, or use JSON with schemas.enable=true

Either way, you need to re-configure the producer of the data

Or, use KSQL to impose a schema on the JSON/CSV data and reserialise it to Avro 👍🏻

Schema, where art thou?

JsonDeserializer with schemas.enable requires "schema" and "payload" fields and may not contain additional fields



Schema

(Embedded per JSON message)

Messages

(JSON)

```
{
  "schema": {
    "type": "struct",
    "fields": [
      {
        "type": "int32",
        "optional": false,
        "field": "id"
      },
[ ... ]
    ],
    "optional": true,
    "name": "asgard.demo.O
  },
  "payload": {
    "id": 611,
    "order_id": 111,
    "customer_id": 851,
    "order_total_usd": 182
    "make": "Kia",
    "model": "Sorento",
    "delivery_city": "Swin
```

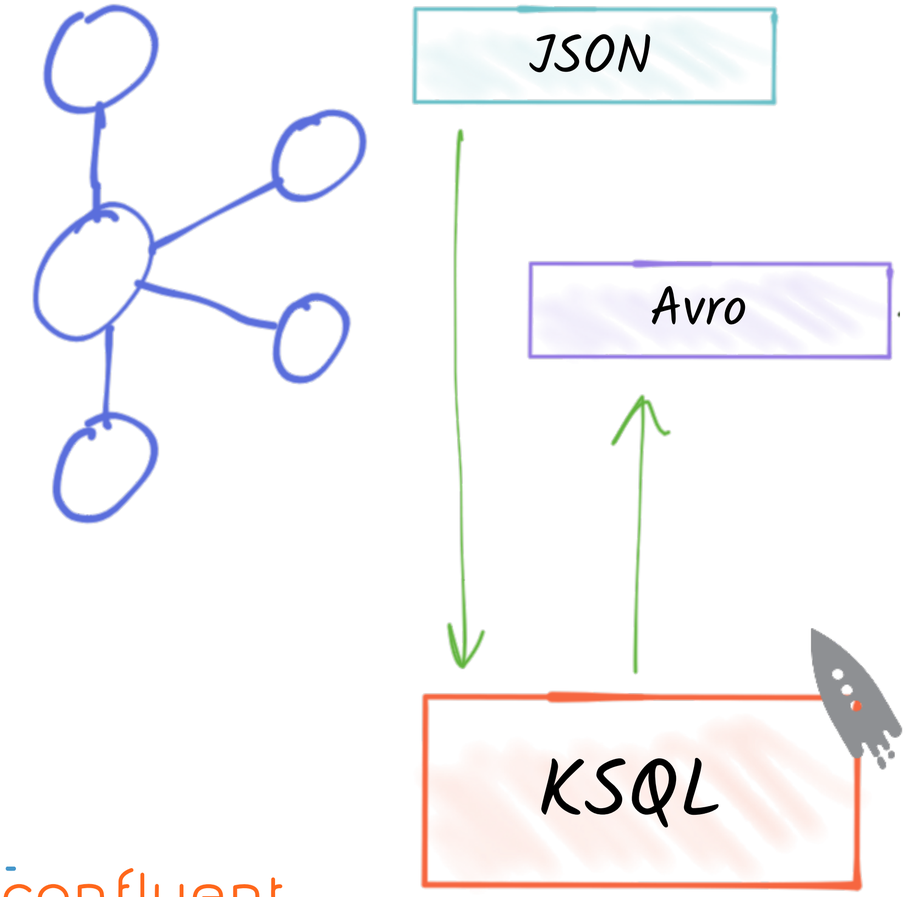
Schema, where art thou?

JsonDeserializer with schemas.enable requires "schema" and "payload" fields and may not contain additional fields

-> Your JSON must be structured as expected.

```
{
  "schema": {
    "type": "struct",
    "fields": [
      {
        "type": "int32",
        "optional": false,
        "field": "id"
      },
[ ... ]
    ],
    "optional": true,
    "name": "asgard.demo.ORDERS
  },
  "payload": {
    "id": 611,
    "order_id": 111,
    "customer_id": 851,
    "order_total_usd": 182190.9
```

Using KSQL to apply schema to your data

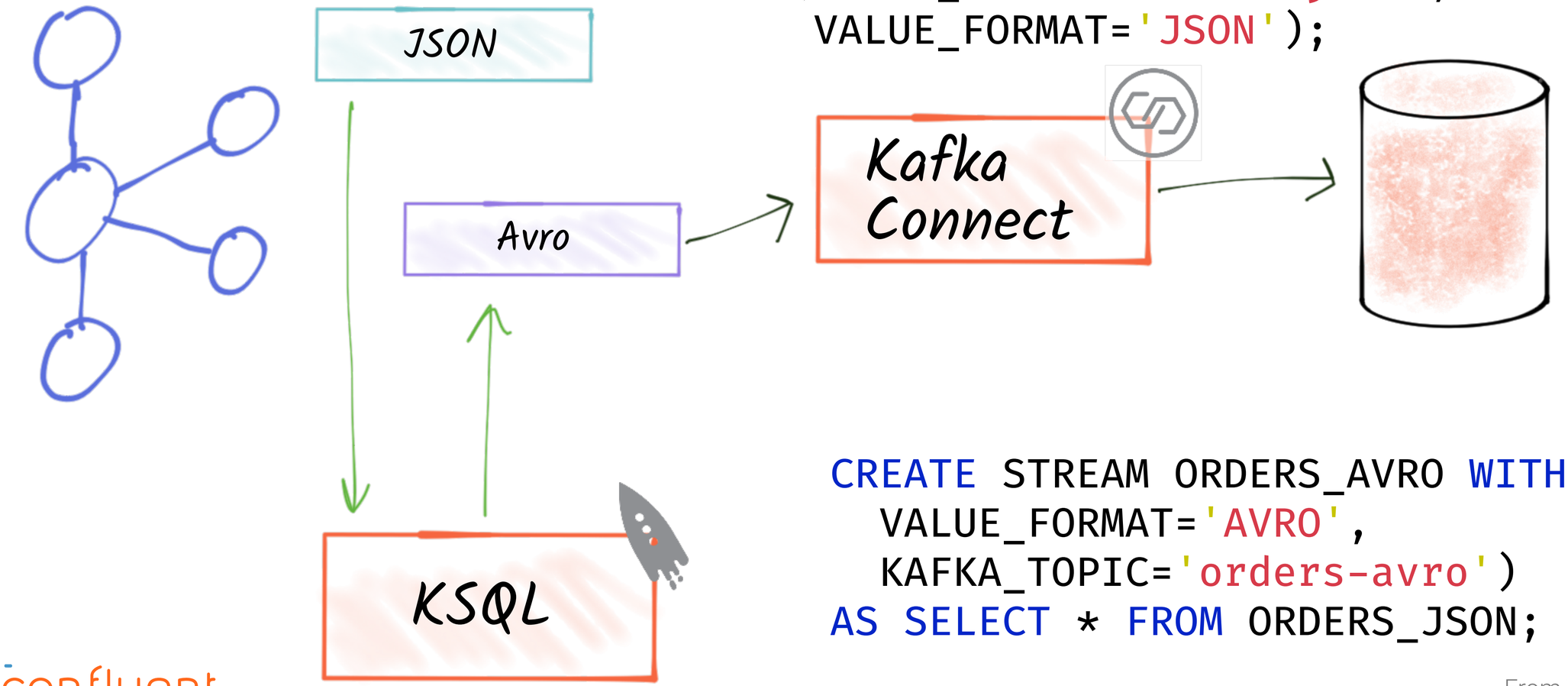



```
CREATE STREAM ORDERS_JSON
  (id INT,
   order_total_usd DOUBLE,
   delivery_city VARCHAR)
  WITH (KAFKA_TOPIC='orders-json',
        VALUE_FORMAT='JSON');
```

```
CREATE STREAM ORDERS_AVRO WITH (
  VALUE_FORMAT='AVRO',
  KAFKA_TOPIC='orders-avro')
AS SELECT * FROM ORDERS_JSON;
```

Using KSQL to apply schema to your data

```
CREATE STREAM ORDERS_JSON
  (id INT,
   order_total_usd DOUBLE,
   delivery_city VARCHAR)
  WITH (KAFKA_TOPIC='orders-json',
        VALUE_FORMAT='JSON');
```



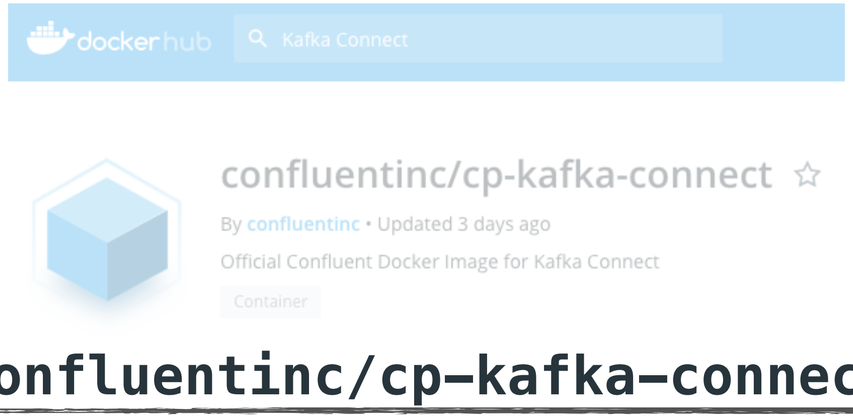

```
CREATE STREAM ORDERS_AVRO WITH (
  VALUE_FORMAT='AVRO',
  KAFKA_TOPIC='orders-avro')
AS SELECT * FROM ORDERS_JSON;
```

Containers

Kafka Connect images on Docker Hub

confluentinc/cp-kafka-connect-base

kafka-connect-elasticsearch
kafka-connect-jdbc
kafka-connect-hdfs
[…]



confluentinc/cp-kafka-connect

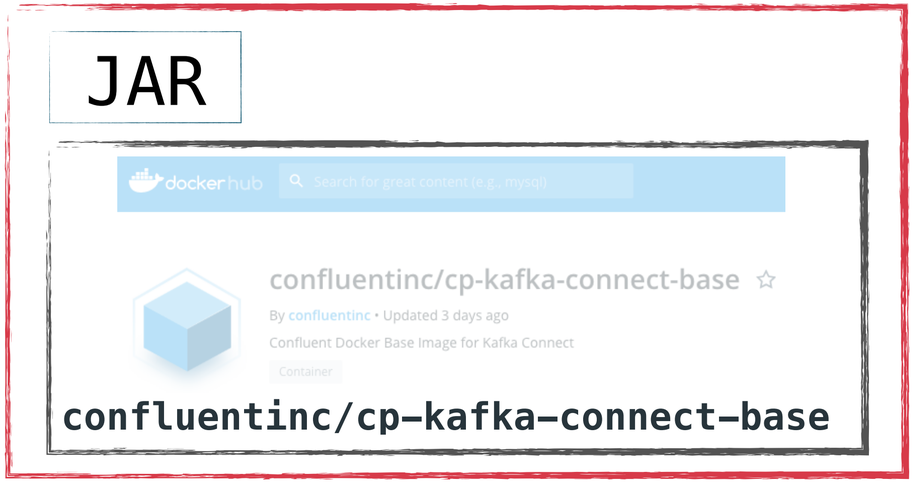
Adding connectors to a container

Confluent Hub

JAR

At runtime

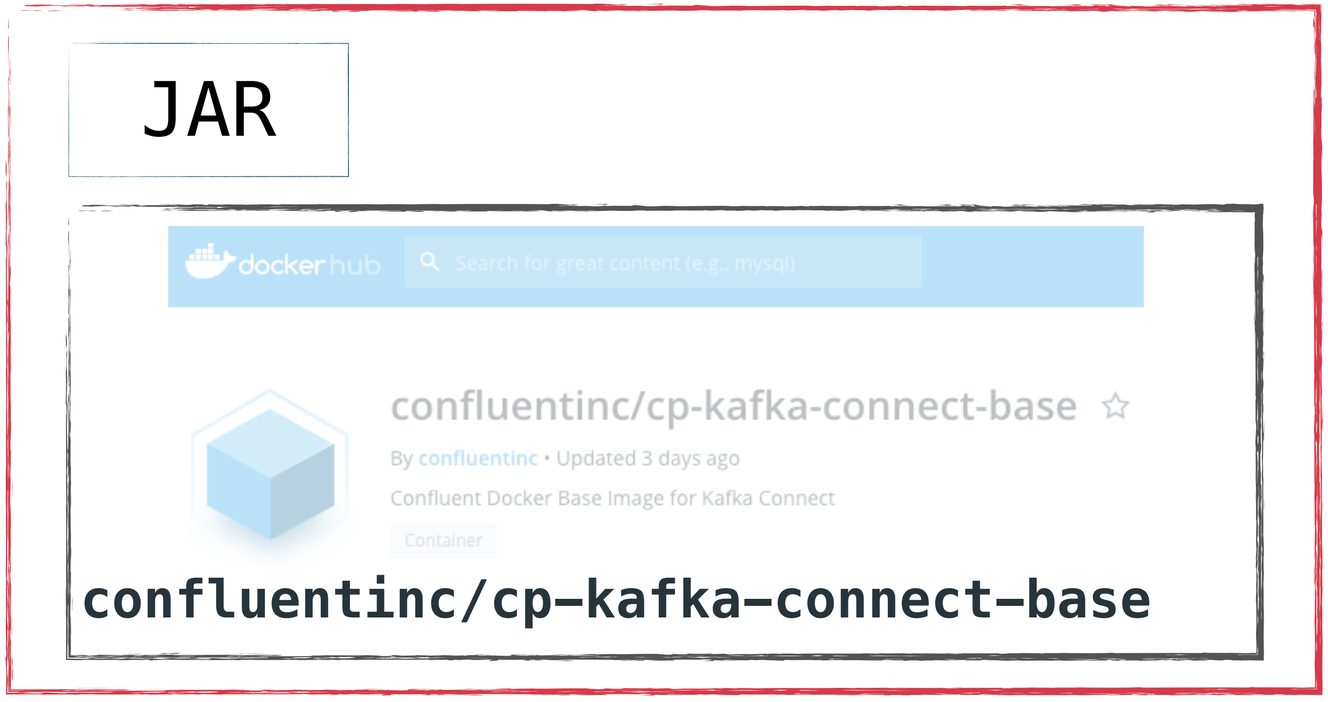
```
kafka-connect:
  image: confluentinc/cp-kafka-connect:5.2.1
  environment:
    CONNECT_PLUGIN_PATH: '/usr/share/java,/usr/share/confluent-hub-components'
command:
    - bash
    - -c
    - |
      confluent-hub install --no-prompt neo4j/kafka-connect-neo4j:1.0.0
      /etc/confluent/docker/run
```



http://rmoff.dev/ksln19-connect-docker

Build a new image

```
FROM confluentinc/cp-kafka-connect:5.2.1
ENV CONNECT_PLUGIN_PATH="/usr/share/java,/usr/share/confluent-hub-components"
RUN confluent-hub install --no-prompt neo4j/kafka-connect-neo4j:1.0.0
```



Automating connector creation

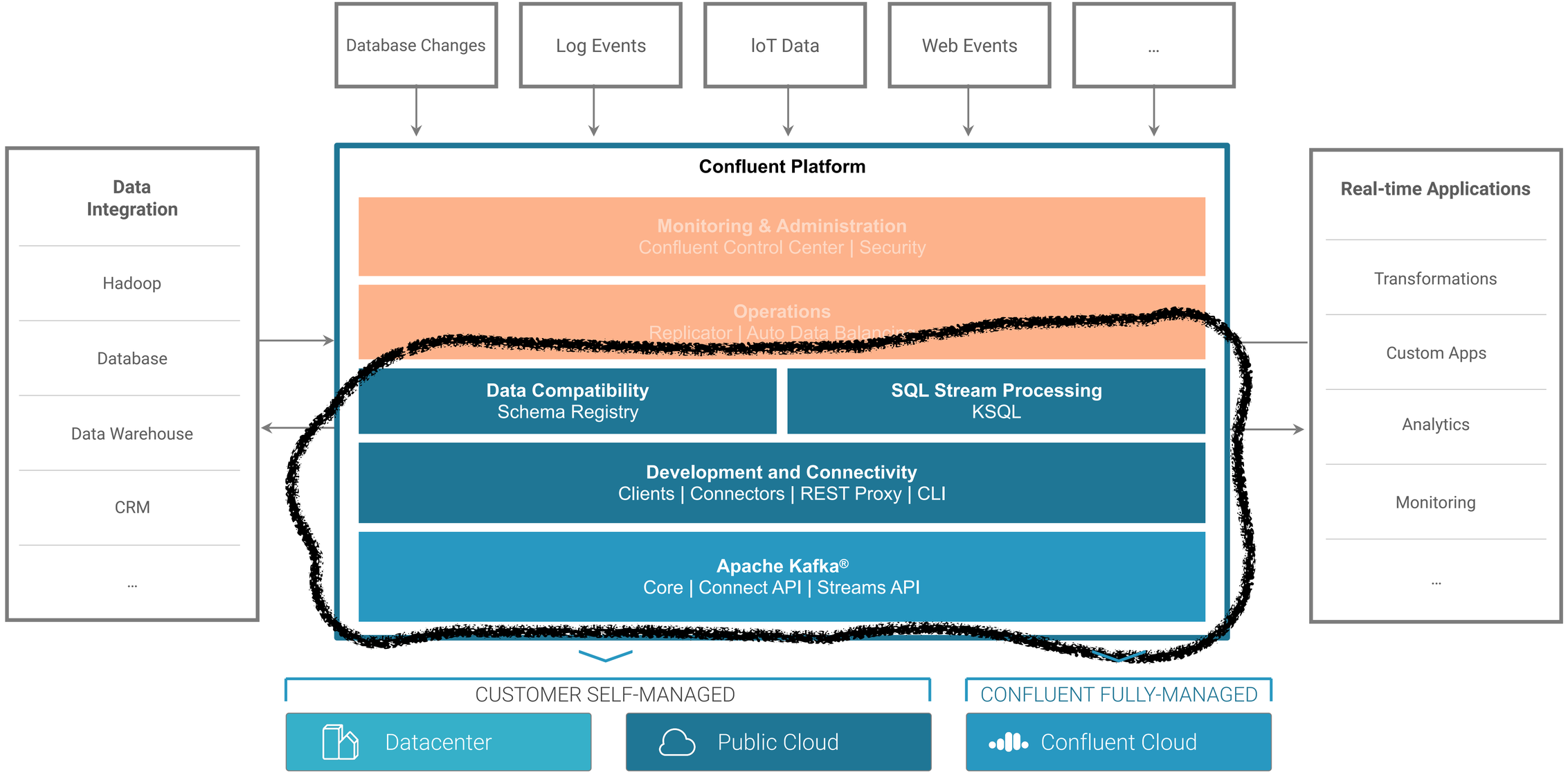
```
# # Download JDBC drivers
cd /usr/share/java/kafka-connect-jdbc/
curl https://cdn.mysql.com/Downloads/Connector-J/mysql-connector-java-8.0.13.tar.gz | tar xz
#
# Now launch Kafka Connect
/etc/confluent/docker/run &
#
# Wait for Kafka Connect listener
while [ $$(curl -s -o /dev/null -w %{http_code} http://$$CONNECT_REST_ADVERTISED_HOST_NAME:$…
  echo -e $$(date) " Kafka Connect listener HTTP state: " $$(curl -s -o /dev/null -w %{http_…
  sleep 5
done
#
# Create JDBC Source connector
curl -X POST http://localhost:8083/connectors -H "Content-Type: application/json" -d '{
          "name": "jdbc_source_mysql_00",
          "config": {
                  "connector.class": "io.confluent.connect.jdbc.JdbcSourceConnector",
                  "connection.url": "jdbc:mysql://mysql:3306/demo",
                  "connection.user": "connect_user",
                  "connection.password": "asgard",
                  "topic.prefix": "mysql-00-",
                  "table.whitelist" : "demo.customers",
                  }
          }'
# Don't let the container die
sleep infinity
```

http://rmoff.dev/ksln19-connect-docker

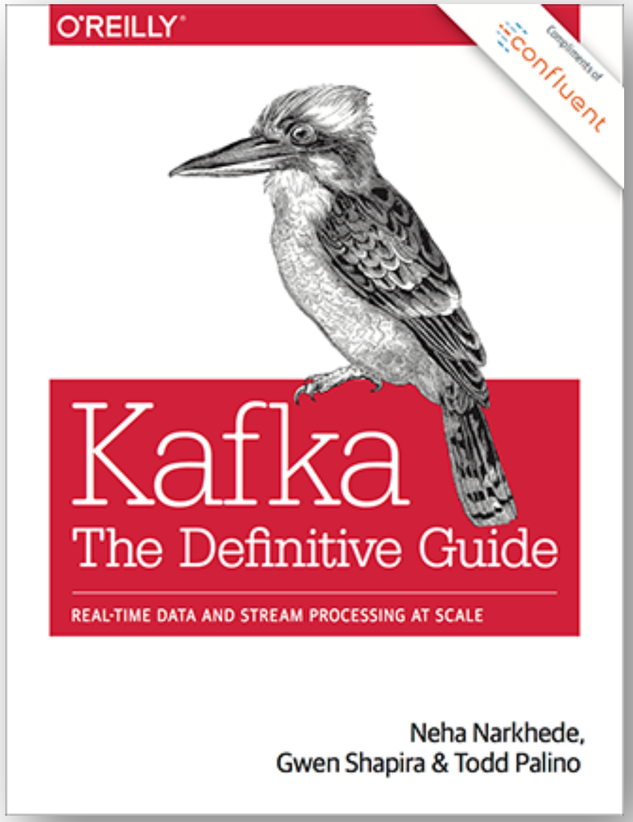
#EOF

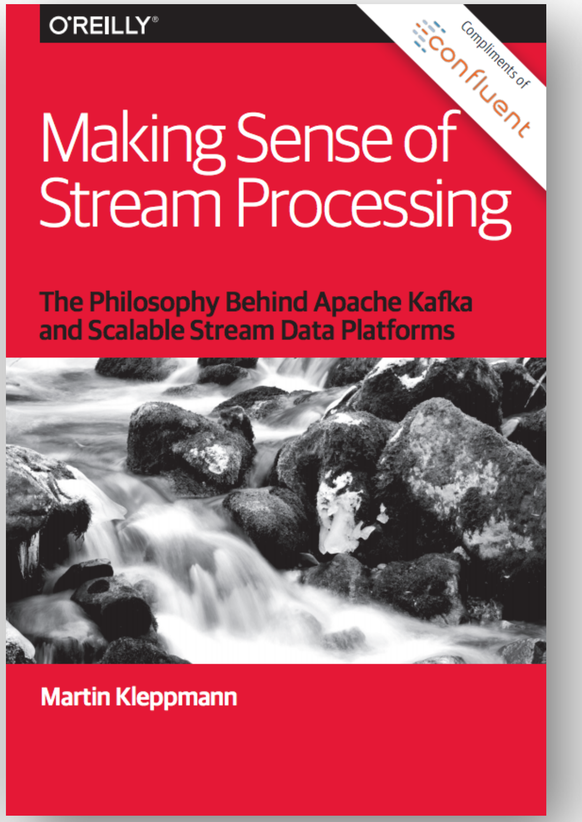
Confluent Community Components

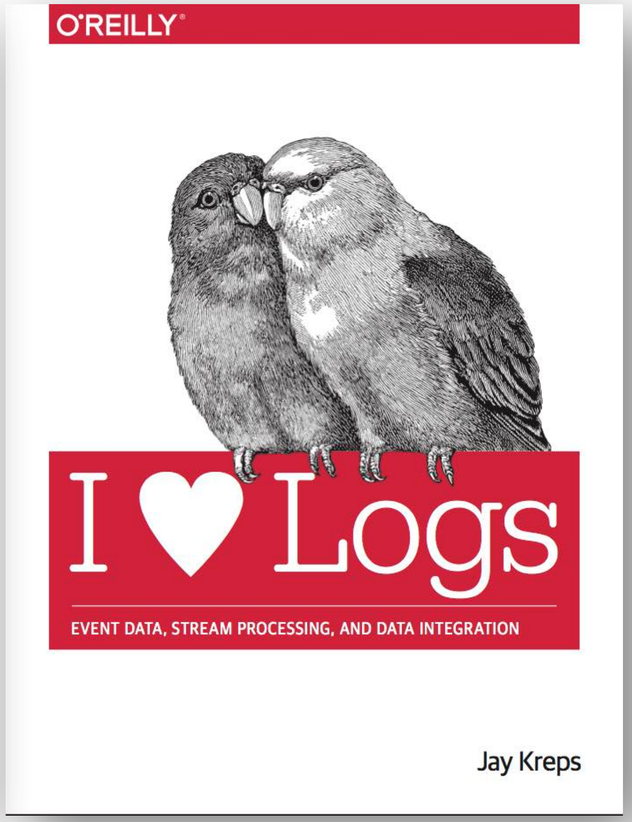
Apache Kafka with a bunch of cool stuff! For free!

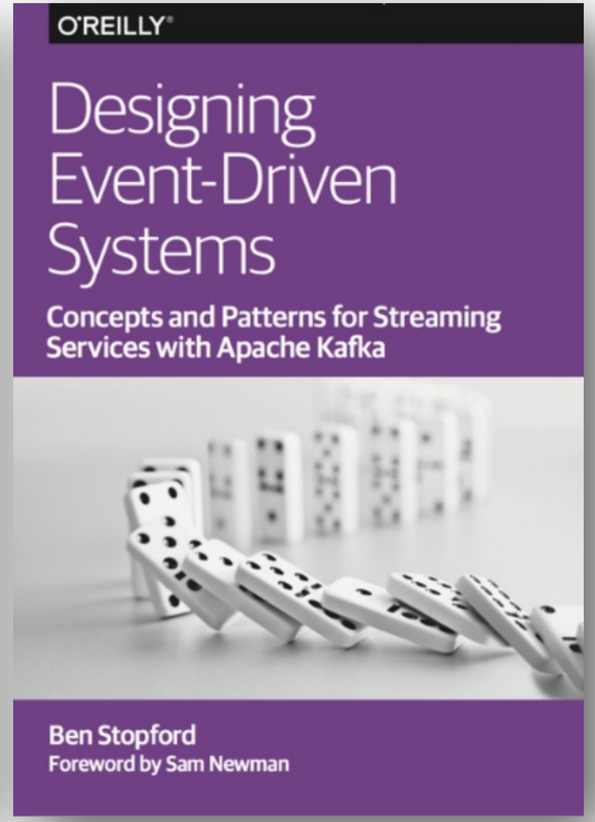


http://cnfl.io/book-bundle









@rmoff
#kafkasummit
Please rate my session in the Kafka Summit app :-)
http://cnfl.io/slack